TRANSMIXR

Ignite the Immersive Media Sector by Enabling New Narrative Visions

D3.2 eXtended Reality Media Experience Format

LEAD AUTHORS:
Aljosa Smolic (HSLU), Lam Kit Yung (HSLU)

AUTHORS/ORGANISATIONS:
CERTH, CWI, TUS, SATORE, MOD, IMM, TCD, Intel

Funded by
the European Union

DELIVERABLE INFORMATION	
Deliverable leader	HSLU
Document type	R
Document name	D3.2 eXtended Reality Media Experience Format
Work Package / Task	WP3 / T3.1 eXtended Reality Media Experience Format
Delivery Date (DoA)	M30 (31st of March 2025)
Actual Delivery Date	31st of March 2025
Authors	HSLU
Reviewers	Intel, TCD

DELIVERABLE HISTORY			
Date	**Version**	**Author**	**Summary of main changes**
30-06-2023	0.1	Aljosa Smolic, Lam Kit Yung	Initial version of chapter 2 (initial review and assessment) delivered to project partners
01-03-2025	0.2	Aljosa Smolic	Structure and content defined for final version
10-03-2025	0.2	All authors	Input provided
19-03-2025	1.0	Aljosa Smolic	First integrated document provided for review
25-03-2025	1.1	John Dingliana (TCD), Björn Browatzki (Intel)	Comments and suggestions received from reviewers
28-03-2025	1.2	Aljosa Smolic	Revised based on reviewer's feedback
31-03-2025	2.0	Aljosa Smolic	Deliverable ready for submission to the EC

DISSEMINATION LEVEL		
PU	Public	**X**
PP	Restricted to other programme participants (including the EC services)	
RE	Restricted to a group specified by the consortium (including the EC services)	
CO	Confidential, only for the members of the consortium (including the EC)	

TRANSMIXR partners

TRANSMIXR

This project has received funding from the Horizon Europe programme under the Grant Agreement 101070109. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the European Commission can be held responsible for them.

Public Executive Summary

Interoperability of systems is crucial for creative and cultural industries. However, in the area of extended (XR) realities, formats and standards are not yet as mature and established as for standard media like video, images, or audio. Therefore, the main objective of T3.1 was to ensure that TRANSMIXR use cases and technologies are sufficiently supported in terms of formats and standards.

Work in T3.1 started early in the project with an initial overview and assessment of XR-related formats and standards. This gave initial recommendations for design and development. At a platform level, OpenXR was identified as the most promising format for general XR experiences. For web applications, WebXR was recommended. At asset level, it was found that many types of content are already well-supported, however this space seems quite fragmented with several specifications for similar purposes. USD and glTF were identified as promising developments going forward, and this is also reflected in industry activities.

A need and opportunity was identified with as volumetric video not yet supported in glTF. To address this, the project developed an extension to glTF named VVglTF. This was published and submitted to a standards organization (Khronos). It was also used in TRANSMIXR's experiences Planica and Performance, and in a follow up project of HSLU. USD was evaluated with the nVidia Omniverse platform in the Performance use case. While USD proved very useful at an asset level, Omniverse is not yet mature enough on platform level.

A variety of XR experiences were developed in TRANSMIXR, including both VR and MR. All of them used OpenXR as their platform format, following the initial assessment and establishing it as TRANSMIXR's "eXtended Reality Media Experience Format". Meta Quest 3, in standalone and tethered modes, was selected as the main hardware platform. Unity and Unreal were applied as game engines.

At an asset level, the experiences use a variety of different 2D and 3D media in related formats, which results in a rich and insightful overall picture. Volumetric video is used in all experiences. VVglTF is used for offline cases, while real-time streaming is done using the VR2Gather system. Standardized real-time streaming of volumetric video is an area of opportunity for further development.

Other, more standard media (3D objects, animations, mocap, avatars, 2D video, 360 video, images, audio, etc.) are well supported by existing formats and standards, but fragmentation is evident in this context. USD and glTF have the potential to streamline and unify this space in the future.

Gaussian Splatting, as a novel type of 3D asset, has a lot of potential for future work, including in the context of formats and standards. For instance, efficient compression is an active area of research.

Overall, we conclude that the objectives related to XR formats and standards could be achieved and that generally the field is maturing, while challenges and opportunities remain.

Table of Contents

List of Figures

List of Tables

1 Introduction

Interoperability of systems is crucial for creative and cultural industries. Creative content passes through pipelines where it is created, edited, authored, composited, transmitted, consumed, etc. Various tools and workflows are used for these operations. Therefore, it is essential that efficient formats and standards (F&S) exist that allow exchange of creative content between platforms and systems. For traditional media this is all well-established. For instance, JPEG is a common standard for digital images that can probably be understood by almost any image-related system in the world today.

In the area of eXtended Reality (XR) F&S are not that well-established yet. A variety of different platforms, often proprietary, are used. Game engines like Unity or Unreal play a central role but cause restrictions. Properties of headsets also bring limitations. A multitude of software tools are available for certain tasks. Different F&S are available for 3D content, while none is so far established like JPEG for images. Different groups and organizations are actively aiming to improve interoperability in the area of XR, while these still have to be brought together and combined efficiently.

In this context, the main objective of this task 3.1 was to **ensure that the requirements regarding F&S of the use cases, pilots and technologies, which were developed in TRANSMIXR, were supported efficiently**. Further, the goal was to identify gaps and opportunities, where requirements were not supported efficiently, to develop related solutions, and to make these solutions available to the project and wider industry, thus creating impact.

In order to get an **initial overview of the state-of-the-art** in the field of F&S relevant for XR, HSLU, with input from partners, performed a review of available platforms as well as 3D asset file formats, along with an initial assessment of the relevance for TRANSMIXR. This overview and assessment was summarized in a preliminary version on D3.2, which was delivered to partners early in the project in order to inform their design and development of technologies and use cases. The initial assessment was updated until final delivery here. This final version can be found in the **section 2 of this report**.

The initial assessment concluded that glTF seems most promising as a 3D file format in the future (see in more detail in section 2). However, it did not support

usage of volumetric video, which is essential for the TRANSMIXR use cases. Thus, we identified an opportunity to develop an extension to glTF for support of volumetric video, VVglTF. Further, the Universal Scene Description (USD) is much more than a file format. It is rather a platform for collaborative, distributed 3D content production, and as such is very interesting for XR content creation on the production side. However, concrete use cases in creative industries are still rare. We therefore identified the opportunity to evaluate usage of USD in the context of the use case Performance. Details about **VVglTF and evaluation of USD** are given in **section 3**.

In parallel to these developments in T3.1, partners worked on the specifications and implementations of the pilots of TRANSMIXR. These are related to the overall architecture principles as reported in D3.1. Different subsets of F&S were applied in the pilots on platform and asset level. In **section 4,** we report the **details of F&S as used in the pilots**.

Finally, **section 5** gives a **summary of learnings and findings** regarding F&S in TRANSMIXR, as well as recommendations and future R&D opportunities.

2 Initial Review and Assessment of XR-related Formats and Standards

In this section, we give a review of XR-related formats and standards and their assessment regarding the relevance for TRANSMIXR. An initial version of this section was provided to partners early in M9 to inform their design and implementation decisions. For final delivery this was updated to the latest status.

We distinguish between the platform level and the asset level. Platforms are frameworks for XR applications, while assets are related to particular media. We focused an 3D assets as formats and standards for standard media such as images or video are sufficiently established.

2.1 Platform Level

In the following subsections, we review and assess XR platforms and their relevance for TRANSMIXR.

2.1.1 OpenXR

2.1.1.1 Description

OpenXR is an open, royalty-free standard developed by the Khronos Group to provide unified access to virtual reality (VR) and augmented reality (AR) platforms and devices. It serves as a cross-platform application programming interface (API). OpenXR enables applications to reach a wider array of hardware platforms without having to port or re-write their code and subsequently allowing platform vendors supporting OpenXR access to more applications.

Currently conformant OpenXR platforms are:

- Microsoft HoloLens 2 and the Windows Mixed Reality headsets
- Oculus PC platform and the Quest /Quest 2 / Quest 3 devices, with full support OpenXR 1.0 added in July 2021.
- Collabora Monado Runtime for GNU/Linux, with the release of version 21.0.0 in February 2021
- Valve SteamVR, since version 1.16 in February 2021
- HTC VIVE Cosmos and VIVE Focus 3, part of HTC's VIVERSE ecosystem
- Qualcomm Snapdragon Spaces XR Developer Platform

2.1.1.2 Relevance for TRANSMIXR

TRANSMIXR focuses on developing immersive media experiences, which could benefit from OpenXR's capabilities for cross-platform VR and AR development. In TRANSMIXR, OpenXR has been adopted in all use cases. In the project, content developers use OpenXR uniform API and SDK to enable sustainable and reusable content development. The OpenXR standard facilitates collaboration between system developers and content developers.

2.1.2 WebXR

2.1.2.1 Description

WebXR is a web application programming interface (API) that enables developers to create immersive virtual reality (VR) and augmented reality (AR) experiences accessible directly through web browsers. It allows web applications to interact with VR and AR hardware, such as head-mounted displays and motion controllers, providing users with seamless immersive experiences without the need for

additional software installations. WebXR enables rapid development of VR/AR experiences in web browsers with the integration of 3D graphics libraries such as A-frame.js and three.js. Web based metaverse platforms such as Mozilla Hubs and Spatial.io provide instant VR immersive experiences in their website.

WebXR Device API supported browsers:

- Chrome, Edge, Safari. 3.1 - 12.1 supported. 13 - 16.2.
- Firefox. 2 - 76 supported. 77 - 108.
- Opera. 10 - 51 supported. 52 - 65
- IE. 6 - 10 supported.
- Chrome for Android. 109.
- Safari on iOS * 3.2 - 16.2 supported. 16.3 supported.

Key Features of WebXR:

- **Cross-Platform Compatibility:** WebXR provides a unified API that works across various devices and platforms, allowing developers to write code once and deploy it universally.
- **Integration with Web Technologies:** Being a web-based API, WebXR integrates smoothly with existing web technologies like HTML, CSS, and JavaScript, facilitating the creation of rich, interactive experiences.
- **Access to Device Capabilities:** WebXR enables direct access to VR and AR hardware features, such as tracking user movements and rendering stereoscopic visuals, enhancing the realism of immersive applications.

2.1.2.2 Relevance for TRANSMIXR

TRANSMIXR could leverage WebXR to:

- **Deliver Immersive Content via the Web:** By utilizing WebXR, TRANSMIXR can provide users with immediate access to immersive experiences through standard web browsers, eliminating the need for additional software installations.
- **Ensure Broad Accessibility:** The cross-platform nature of WebXR allows TRANSMIXR to reach a wider audience, as users can access content on various devices, including desktops, smartphones, and VR headsets.
- **Simplify Development and Deployment:** Integrating WebXR can streamline the development process by allowing TRANSMIXR to use standard web

technologies, reducing complexity and facilitating faster deployment of immersive applications.

Demostrators viewable in WebXR supported web browsers could be distributed to the public via URLs. User study participants and reviewers could easily access new content without the need for complicated installation of VR experiences. Content creators could use WebXR to develop XR content along with text-based content on their websites. WebXR could effectively enhance the ability to create and distribute immersive media experiences, aligning with objectives to innovate in the realms of VR and AR storytelling.

2.1.3 NVIDIA Omniverse

2.1.3.1 Description

NVIDIA Omniverse is a real-time 3D graphics collaboration platform for creating and operating metaverse applications on systems equipped with NVIDIA display cards. It has been used for applications in the visual effects and "digital twin" industrial simulation industries. Omniverse makes extensive use of the Universal Scene Description (USD) format. Omniverse enables seamless interoperability between various 3D software applications, allowing designers, engineers, and creators to work together in a unified virtual environment.

NVIDIA Omniverse Kit is the SDK on which every Omniverse microservice (like DeepSearch) or reference application (such as Omniverse Create, View, or Isaac Sim) is built.

Key Features of NVIDIA Omniverse:

- **Real-Time Collaboration:** Multiple users can simultaneously work on the same 3D scene, with changes reflected instantly across all connected applications.
- **Photorealistic Rendering:** Leveraging NVIDIA's RTX technology, Omniverse provides high-fidelity, real-time rendering capabilities, enabling users to visualize complex scenes with accurate lighting and materials.
- **Extensible Platform:** Through connectors, Omniverse integrates with popular 3D content creation tools such as Autodesk Maya, Blender, and Unreal Engine, ensuring a flexible and adaptable workflow.

- **Physics Simulation:** Omniverse incorporates NVIDIA's PhysX engine to simulate realistic physical interactions within virtual environments, essential for applications in robotics, autonomous vehicles, and industrial simulations.

2.1.3.2 Relevance for TRANSMIXR

NVIDIA Omniverse could be a good internal asset management platform for TRANSMIXR. It enables high quality 3D asset development and publishing.

- **Enhance Collaborative Content Creation:** By providing a real-time collaborative environment, Omniverse can facilitate the joint development of immersive media experiences among distributed teams, aligning with TRANSMIXR's goals of advancing storytelling through technology.
- **Streamline Asset Integration:** The interoperability offered by Omniverse allows for the seamless incorporation of assets from various design tools, potentially simplifying workflows and improving efficiency within the TRANSMIXR project.
- **Simulate Interactive Scenarios:** With robust physics simulation capabilities, Omniverse can be employed to create and test interactive scenarios, contributing to the development of engaging and realistic immersive experiences.

By leveraging NVIDIA Omniverse, TRANSMIXR could potentially enhance its ability to create, simulate, and collaborate on complex 3D content, thereby enriching the immersive media experiences it aims to deliver.

2.1.4 SteamVR

2.1.4.1 Description

SteamVR is a virtual reality (VR) hardware and software platform developed by Valve Corporation, designed to facilitate immersive VR experiences across a range of headsets and devices. It is a VR system based on Open Source Virtual Reality libraries. It provides a platform for VR devices including Valve Index, HTC Vive, Oculus Rift, Windows Mixed Reality headsets. SteamVR runs as a part of the Steam client. Game engines such as Unreal Engine and Unity 3D communicate with VR hardware through the SteamVR system.

2.1.4.2 Relevance for TRANSMIXR

SteamVR could be utilized to:

- **Develop Immersive Experiences:** Leverage room-scale VR capabilities to create engaging storytelling environments.
- **Ensure Hardware Compatibility:** Utilize SteamVR's broad headset support to reach a wide audience.
- **Enhance User Interaction:** Incorporate features like the Chaperone system to improve user safety and immersion.

By integrating SteamVR, content developers could effectively deliver rich, immersive media experiences across diverse VR platforms. SteamVR provides better user experience than OpenXR in motion tracking for Steam powered VR devices. Developers are advised to use SteamVR for Steam powered VR devices.

2.1.5 MRTK

2.1.5.1 Description

Microsoft's Mixed Reality Toolkit (MRTK) is a cross-platform toolkit that accelerates cross-platform MR app development for Virtual Reality (VR) and Augmented Reality (AR).

MRTK enables Rapid Prototyping, which is made possible by the many User Interface (UI) and User Experience (UX) tools, prefabs (templates of objects, like buttons), and examples that MRTK comes with, out of the box such as:

- Near and Far Interactions for more intuitive user navigation.
- Subtle use of Solvers to keep pertinent information within the user's field of view without overwhelming them.
- Consolidated Hand Menus to free up precious XR screen space without losing functionality.

MRTK also enables faster UI/UX Experience Polish with access to MRTK's "basic" building blocks and core systems. This allows teams to work in a "Design, Test, Polish, Repeat" pipeline that opens up a large amount of flexibility while keeping the base of the application solid and stable.

2.1.5.2 Relevance for TRANSMIXR

With MRTK, we can get an idea up and running, polish it based on testing, and put out a finished application; all within a very short timeframe. Being able to use a single device, the HoloLens 2 for instance, to test for a large portion of the core development process saves time. Developer can use MRTK's cross-platform support to publish to multiple devices, for final polish and testing.

2.1.6 AR Foundation

2.1.6.1 Description

AR Foundation is a cross-platform framework developed by Unity Technologies that enables developers to build augmented reality (AR) applications. It provides a unified API to work seamlessly across various AR platforms, including ARKit (iOS) and ARCore (Android), allowing developers to write code once and deploy it on multiple devices. AR Foundation is only available with the Unity 3D game engine.

2.1.6.2 Relevance for TRANSMIXR

The TRANSMIXR project aims to create distributed XR environments for immersive storytelling, encompassing both AR and VR content creation and delivery. It's plausible that AR Foundation could be utilized within TRANSMIXR to:

- **Develop AR Experiences:** Enable the creation of interactive AR applications that integrate seamlessly with existing platforms.
- **Ensure Cross-Platform Compatibility:** Facilitate the deployment of AR content across various devices, ensuring broad reach.
- **Streamline Development:** Leverage AR Foundation's unified API to expedite the development process and maintain consistency across different AR platforms.

By incorporating AR Foundation, TRANSMIXR could effectively streamline the development and distribution of immersive AR experiences, aligning with its objectives to enhance storytelling through advanced technologies.

2.1.7 Unity XR SDK

2.1.7.1 Description

Unity XR SDK is Unity's framework is designed to streamline the development of Virtual Reality (VR) and Augmented Reality (AR) applications. It provides a unified interface, allowing developers to create immersive experiences that are compatible across various XR platforms without the need for platform-specific code.

Key Features of Unity XR SDK:

- **Unified API:** Offers a consistent set of interfaces for different XR devices, simplifying the development process.
- **Extensibility:** Allows for the integration of new XR devices and platforms through custom plugins.
- **Cross-Platform Compatibility:** Supports a wide range of XR platforms, including Oculus, PlayStation VR, ARKit, ARCore, Windows Mixed Reality, and more.

Unity XR SDK is aimed at specialist users who want to develop their own XR providers that work with Unity. The XR SDK package allows multiple backends (called “providers”) to implement a single engine feature (called a “subsystem”) in Unity. User applications can select and activate providers at runtime.

2.1.7.2 Relevance for TRANSMIXR

TRANSMIXR may leverage the Unity XR SDK to develop cross-platform immersive experiences. By utilizing Unity's XR framework, content developers can efficiently create applications that operate seamlessly across various VR and AR devices, aligning with its goal to enhance immersive storytelling and media experiences. Using the Unity XR SDK enables content developers and users to adopt hardware with minimal effort during the development process.

2.1.8 Unity XR Interaction Toolkit

2.1.8.1 Description

The XR Interaction Toolkit is a high-level component-based interaction system for developing VR and AR experiences. It provides a framework for integrating 3D and UI interactions with Unity input events. The core system consists of a collection of Interactor and Interactable components, as well as an Interaction Manager that

connects these two types of components. It also includes components for movement and the creation of visuals. XR Interaction Toolkit contains a set of components that support the following Interaction tasks:

- Cross-platform XR controller input: Meta Quest (Oculus), OpenXR, Windows Mixed Reality or more
- Basic object hover, select and grab
- Haptic feedback through XR controllers
- Visual feedback (tint/line rendering) to indicate possible and active interactions
- Basic canvas UI interaction with XR controllers
- Utility for interacting with XR Origin, a VR camera rig for handling stationary and room-scale VR experiences

2.1.8.2 Relevance for TRANSMIXR

The Unity XR Interaction Toolkit provides templates for simple interaction with virtual objects. It helps developers to achieve consistent interaction experience with different VR devices, through its API. Users will be able to adopt most of the supported devices to present their interactive contents in the metaverse.

2.1.9 Unreal Engine 5

2.1.9.1 Description

Unreal Engine 5 (UE5) is a cutting-edge game engine developed by Epic Games, designed for high-fidelity real-time 3D rendering. It is widely used in gaming, virtual production, XR (Extended Reality), and simulations due to its powerful rendering capabilities, physics engine, and AI integration.

Key Features of Unreal Engine 5:

- Real-time dynamic lighting for photo-realistic XR scenes, crucial for immersive cultural heritage reconstructions.
- MetaHuman Creator: Generates ultra-realistic digital humans for VR/AR applications. Useful for TRANSMIXR's use cases, enabling virtual anchors, avatars, or performers.
- Motion Capture & Virtual Production: Supports Live Link Face, Rokoko, Xsens, and other MoCap systems. Enables real-time interaction with 3D assets, useful for performing arts in TRANSMIXR.

- XR & Metaverse Support. Native integration for VR/AR/MR development. Compatible with Oculus, HTC Vive, and ARKit/ARCore.
- World Partition System: Handles large-scale, open-world environments efficiently. Ideal for creating interactive cultural heritage and immersive storytelling.

2.1.9.2 Relevance for TRANSMIXR

Unreal Engine 5 is widely adopted in the professional production industry due to its high quality features and performance. In TRANSMIXR the use case Performance bases their development on this platform (see below).

2.2 Asset Level

In the following we review and assess 3D asset formats and their relevance for TRANSMIXR.

2.2.1 glTF

2.2.1.1 Description

glTF is a file format for publishing 3D assets (3D model geometry, appearance, scene graph hierarchy, and animation.) on the Internet. It relies on open standards and integrated browser technologies such as WebGL. It can efficiently stream large and highly detailed 3D models over the Internet. glTF contents are encapsulated in JSON format and contain direct instruction to WebGL renderers. Thus, it meets the rendering performance requirements in WebGL-enabled web browsers. glTF supports Physically Based Rendering 3D materials and enables developers and artists to achieve photorealism through rendering parameters that correspond to real-world physical properties of materials in 3D assets.

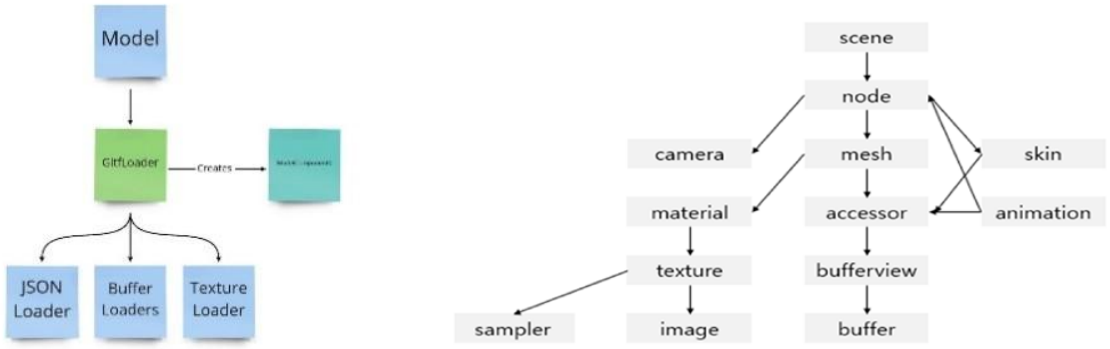

Figure 1: Left: The glTF loader pipeline. Right: The file structure of glTF.

glTF™ 2.0 has been released as the ISO/IEC 12113:2022 International Standard. Khronos completed the transposition of glTF 2.0 through the ISO/IEC JTC 1 PAS (Publicly Available Specification) process to solidify glTF's global recognition and accelerate its adoption by industry and other standards. Khronos will continue to evolve glTF and will regularly update ISO/IEC 12113 with proven, widely available glTF functionality to avoid industry fragmentation. glTF extensions expand the functionality of the underlying glTF model format. Extensions provide the capability to add novel attributes, such references to other data. Additionally, extensions can describe the structure and format of this external data. Furthermore, extensions can offer new meanings for parameters, establish reserved identification numbers, and create fresh container formats. Extensions have the potential to be included into the core glTF specification in a subsequent version of glTF.

2.2.1.2 Relevance for TRANSMIXR

glTF could be employed to streamline the integration and deployment of 3D assets across various platforms and devices. This would align with TRANSMIXR's goals of enhancing immersive media experiences by ensuring efficient and consistent delivery of 3D content.

According to our comparison analysis (see below), glTF is the best portable 3D model file format for TRANSMIXR use cases. Collaborative editing of 3D models is possible with the glTF file format. Adapting the glTF format could accelerate the delivery of 3D contents, from 3D model creation to publishing.

2.2.2 VVglTF

2.2.2.1 Description

VVglTF is a custom extension to enable Volumetric Video (VV) playback based on the glTF 2.0 standard (see more detail in section 3). It utilizes the same underlying graph structure of glTF as the core specification. In VVglTF, every frame of a video sequence has a corresponding mesh stored in the glTF file. Texture video provides a texture for every 3D mesh of the VV sequence. Using the VVglTF file format enables using VV with the portable file format and multi-platform support.

VVglTF uses the video player API to decode the video, get the video duration (total number of frames), frame rate and the video frames. The VV controller converts each video frame to a texture and assigns them to the material of the VV meshes. The glTF render engine only renders and shows the node elements specified in the scene description. It ignores the elements missing in the scene structure, even if the elements are embedded in the glTF file. VVglTF uses this feature to enable efficient rendering of VV with glTF. This is controlled by the video, which activates only one 3D mesh at a time and provides the corresponding texture to it. VVglTF keeps all 3D meshes of the VV sequence in the buffer, each VV frame mesh has a unique mesh ID and primitive description data in the glTF file, following the glTF 2.0 specification.

We also developed a simple and efficient VVglTF streaming system based on HTTP Live Streaming technology for video texturing. It is designed to efficiently play VV content across different network conditions by adjusting the frame rate and number of frames per glTF file.

2.2.2.2 Relevance for TRANSMIXR

VVglTF serves as a crucial extension to the glTF file format, tailored to meet the demands of VV by providing efficient data handling, adaptive streaming capabilities, and cross-platform interoperability. Its implementation within the TRANSMIXR project underscores its significance in advancing immersive media technologies.

Within the TRANSMIXR project, VVglTF has been adopted in the Newsroom and the Performance use cases (see more details in section 4). It plays a pivotal role in:

- Standardizing Volumetric Video Formats: The project utilizes the VVglTF extension to establish standardized methods for encoding, decoding, and rendering volumetric video, ensuring consistency and interoperability across different platforms.
- Enhancing Immersive Applications: By integrating VVglTF, TRANSMIXR enables the representation of real people in extended reality (XR) environments, paving the way for novel forms of immersive applications and storytelling.

2.2.3 glXF

2.2.3.1 Description

glTF Experience Format (glXF) is the specification for a new file type in the glTF ecosystem for efficient composition of multiple glTF assets. glXF contains asset lists in URIs to incorporate node-hierarchies or scenes from listed assets. glFX brings more interactive features to the glTF format. glFX enables efficient development of interactive designs with 3D models. It makes it simple to develop interactive 3D models with the glFX format.

Figure 2: Left: The glXF interaction concept, Right: example of glXF.

2.2.3.2 Relevance for TRANSMIXR

In XR applications, managing complex scenes with numerous 3D assets is crucial for creating immersive experiences. The glTFx format addresses this need by enabling:

- **Efficient Scene Composition:** Developers can construct intricate XR environments by referencing existing glTF assets, reducing redundancy and optimizing performance.

- **Level-of-Detail (LoD) Management:** glTFx supports LoD configurations, allowing XR applications to adjust asset details based on factors like user proximity, thereby enhancing rendering efficiency.
- **Interactivity and Anchoring:** The format's extensibility permits the inclusion of interactive elements and augmented reality (AR) anchoring, essential for dynamic XR experiences.

2.2.4 USD

2.2.4.1 Description

Pixar's Universal Scene Description(USD) is an open, extensible framework and ecosystem with APIs for composing, editing, querying, rendering, collaboration, and simulation within 3D virtual worlds. The USD standard is a interchange format that can be used for file editing in Digital Content Creation Tools such as Maya.

The USD file format enables iOS users to produce and view 3D models with good quality. Nvidia Omniverse natively support the USD file format for interoperability within their ecosystem. Developing the metaverse contents in USD could ensure the interoperability across Nvidia and iOS platforms. USD enables non-destructive workflows and collaboration in scene creation and asset aggregation so teams can iterate collaboratively.

USDz is subset of USD, used by Apple and supported in iOS AR applications (a zip file with USD and image files). USDZ delivers AR and 3D content to Apple devices. Apple ARKit and RealityCapture natively support the USD file format with detail texture. The USDZ file format enables the following AR abilities to 3D assets:

- Anchor 3D content at a specific location in the real world.
- React to real-world situations.
- Participate in a physics simulation.
- Connect audio effects to a location.
- Annotate the environment by displaying text.

2.2.4.2 Relevance for TRANSMIXR

USD plays a key role in TRANSMIXR as it is used in the Performance use case as described in more detail in section 3.

2.2.5 MPEG Immersive Video (MIV)

2.2.5.1 Description

MIV provides the capability to compress a representation of a real or virtual 3-D scene captured by multiple real or virtual cameras. The MIV standard enables storage and distribution of immersive video content over the networks, for playback with 6 degrees of freedom (6 DoF) of view position and orientation within a limited viewing space and with different fields of view depending on the capture setup.

The MIV standard can be used in many VR, augmented reality (AR), and mixed reality (MR) use cases. Immersive video playback devices include Head-Mounted Displays (HMDs), holographic displays, or ordinary 2-D displays with input of the viewers' position and orientation.

MIV enables video-based textures to volumetric 3D models in the metaverse. An example use case which benefits from immersive video. A viewer can choose to watch a sporting event from any desired position and orientation. Education and training use cases provide a student with a 3-D view of objects and scenes, seen from a variety of perspectives. Immersive video makes video conferencing/telepresence and virtual tourism more realistic. A news agency could use sports viewing technology to report a match. The V3C Immersive Platform from the 5G-MAG MEDIA ACTION GROUP, provides V3C simple player (Unity application) to visualize volumetric video contents in MIV file format (https://5g-mag.github.io/Getting-Started/pages/v3c-immersive-platform/).

Key Features of MPEG Immersive Video (MIV):

- 6DoF Movement → Users can move left-right, up-down, forward-backward within the scene.
- Depth-based View Synthesis → Uses depth maps to reconstruct new viewpoints dynamically.
- Optimized for Streaming → Compatible with 5G, cloud rendering, and edge computing for real-time streaming.
- Interoperability with Existing Codecs → Uses HEVC (H.265) or VVC (H.266) for efficient compression.
- Supports Point Clouds & Volumetric Video → Works with MPEG Point Cloud Compression (PCC) for integrating 3D objects into immersive video experiences.

File Formats Used in MIV:

- **.mp4** → ISO Base Media File Format (ISOBMFF) for storing immersive video.
- **.miv** → MPEG Immersive Video-specific container format.
- **.bin** → Metadata storage (e.g., depth maps, camera parameters, and reconstruction data).

2.2.5.2 Relevance for TRANSMIXR

MIV is an interesting format for immersive scenes. However, the integration with computer graphics content and pipelines used in XR is limited, which limits usability for TRANSMIXR use cases.

2.2.6 MPEG Point Cloud Compression (MPEG-PCC)

2.2.6.1 Description

Typically, volumetric visual data is computer-generated or captured from the real world. Point cloud is a typical format for representing such data, together with polygonal mesh. Hence, Point Cloud Compression (PCC) is a technique for compressing volumetric visual data. MPEG-PCC is a well-designed point cloud standard (ISO/IEC 23090-5) comprising a list of 3 points coordinate (x, y, z) along with reflectance and RGB attributes associated with each point.

The MPEG Point Cloud Compression (PCC) standard encompasses two primary methodologies:

- **Video-Based Point Cloud Compression (V-PCC):**
  - **Approach:** Transforms 3D point cloud data into 2D video frames, leveraging existing video codecs (e.g., HEVC, AV1) for compression.
  - **Implementation:** The reference software, known as Test Model Category 2 (TMC2), is accessible on MPEG's GitHub repository.
  - V3C Immersive Platform from 5G-MAG MEDIA ACTION GROUP, also provide V3C simple player (Unity application) to visualize volumetric video contents in **V-PCC** file format.
- **Geometry-Based Point Cloud Compression (G-PCC):**
  - **Approach:** Directly compresses 3D point cloud data using spatial coding techniques, such as octree coding and predictive coding.
  - **Implementation:** The reference software, Test Model Category 13 (TMC13), is also available on MPEG's GitHub repository

2.2.6.2 Relevance for TRANSMIXR

MPEG-PCC is relevant for point cloud data produced in TRANSMIXR use cases.

2.2.7 MPEG Dynamic Mesh Coding

2.2.7.1 Description

MPEG Dynamic Mesh Coding (DMC) is a standard developed by the Moving Picture Experts Group (MPEG) to efficiently compress and transmit dynamic 3D mesh sequences. Dynamic meshes, characterized by their time-varying geometry and connectivity, are essential in applications such as immersive media, virtual reality (VR), and augmented reality (AR). Traditional static mesh compression techniques are inadequate for these dynamic models due to their evolving structures over time. To address this, MPEG initiated efforts to standardize DMC, leading to the development of the Video-based Dynamic Mesh Coding (V-DMC) standard.

Key Features of MPEG Dynamic Mesh Coding:

- **Efficient Compression:** DMC employs advanced algorithms to reduce the data size of dynamic meshes, facilitating efficient storage and transmission without significant loss of quality.
- **Support for Time-Varying Connectivity:** Unlike previous standards, DMC accommodates meshes with changing connectivity over time, making it suitable for complex animations and simulations.
- **Integration with Video Codecs:** The V-DMC approach leverages existing video codecs by projecting 3D mesh data onto 2D planes, enabling the use of mature video compression technologies for mesh data.

2.2.7.2 Relevance for TRANSMIXR

Applying MPEG Dynamic Mesh Coding for 3D models offers several advantages to TRANSMIXR use cases:

- Enhanced Immersive Content: By utilizing DMC, TRANSMIXR use cases can incorporate high-quality, dynamic 3D models into its applications, enriching user experiences with realistic animations and interactions.
- Optimized Performance: Efficient compression of dynamic meshes ensures smoother streaming and reduced latency, which are critical for maintaining immersion in VR and AR environments.

- Interoperability: Adherence to MPEG standards facilitates compatibility with a wide range of devices and platforms, allowing TRANSMIXR to reach a broader audience without compatibility issues.

2.2.8 Draco Compression

2.2.8.1 Description

Draco is an open-source library for compressing and decompressing 3D geometric meshes and point clouds. It improves the storage and transmission of 3D assets by significantly reducing the size of 3D assets while maintaining visual fidelity, making it an essential tool for applications requiring efficient 3D model transmission and rendering. Draco is supported by glTF as the built-in compression library. three.js is a JavaScript library to enable 3D rendering in web browsers.

Key Features of Draco Compression:

- **Geometry Compression**: Compresses 3D meshes and point clouds with high efficiency, reducing file sizes dramatically.
- **Lossy and Lossless Compression**: Supports both lossy (higher compression, lower quality) and lossless (preserves all data) compression methods.
- **Faster Transmission and Loading**: Reduces bandwidth and storage requirements, making 3D assets load faster in real-time applications.
- **Cross-Platform Support**: Compatible with WebGL, glTF, and major 3D graphics engines like Unity and Unreal Engine.
- **glTF Integration**: Widely used with **glTF 2.0** to compress 3D assets for web and VR/AR applications.

2.2.8.2 Relevance for TRANSMIXR

In TRANSMIXR use cases Newsroom and Performance, Draco compression has been adopted with VVglTF file, to optimize the storage, transmission, and rendering of long time Volumetric Video. Since AR/VR applications need to maintain real-time performance, using Draco allows TRANSMIXR to reduce memory and processing load without sacrificing quality. This results in **smoother** and **faster** experiences in immersive environments, making it possible to stream volumetric content efficiently over networks with minimal latency.

2.2.9 BVH

2.2.9.1 Description

Biovision Hierarchy (BVH) is a widely used file format for storing **motion capture (MoCap) data**, to represent **skeletal animations**. BVH files contain two key sections:

- **Hierarchy Section**: Defines the skeleton structure, including bones, joints, and their parent-child relationships.
- **Motion Data Section**: Stores rotation and position data for each joint over time, allowing animation playback.

2.2.9.2 Relevance for TRANSMIXR

BVH offers the following features:

- Motion Capture for XR Avatars
  - BVH files enable realistic character animations in VR/AR environments.
  - Avatars in TRANSMIXR can use BVH-based motion tracking to enhance user interaction.
- Live Performance and Virtual Production
  - BVH-based motion data can be integrated into volumetric video and virtual actors, allowing real-time motion capture streaming for immersive storytelling.
- Interoperability with Animation Pipelines
  - Since BVH is a standardized format, it allows TRANSMIXR to integrate motion capture data from different sources into game engines like Unity and Unreal Engine.

2.2.10 PBR Texturing

2.2.10.1 Description

Physically Based Rendering (PBR) texturing is a **realistic** approach to creating materials in 3D graphics. It simulates how light interacts with different surfaces, ensuring accurate reflections, shadows, and material properties. PBR follows principles based on real-world physics, making it the industry standard for modern **game engines, XR experiences, and digital content creation**.

Key Features of PBR Texturing:

- **Physically Accurate Materials** – Simulates real-world material properties like metal, wood, glass, and fabric.
- **Energy Conservation** – Ensures materials do not reflect more light than they receive, maintaining realism.
- **Lighting Independence** – Looks realistic in various lighting conditions, making assets reusable across different environments.
- **Two Main Workflows:**
  - **Metallic-Roughness:** Used in glTF and modern game engines.
  - **Specular-Glossiness:** Offers finer control over reflections, but is less commonly used.

2.2.10.2 Relevance for TRANSMIXR

PBR texturing is useful for creating high-quality, realistic, and interactive content and offers several advantages:

- Enhancing XR Visual Fidelity
  - AR/VR experiences require realistic materials to improve immersion.
  - PBR ensures consistent quality across different lighting conditions in virtual production and XR storytelling.
- Optimized for Real-Time Rendering
  - PBR textures are designed for real-time rendering in Unity, Unreal Engine, and NVIDIA Omniverse, making them ideal for TRANSMIXR's interactive applications.
  - Works well with WebXR and mobile AR to ensure high-quality visuals on limited hardware.
- Interoperability with glTF and Omniverse
  - glTF (Graphics Language Transmission Format), widely used in XR pipelines, natively supports PBR materials.
  - Can leverage NVIDIA Omniverse for collaborative, cloud-based PBR material workflows.
- Realism in Virtual Avatars and Environments
  - PBR enables lifelike characters and environments in volumetric video and 3D assets.
  - Scanned materials (photorealistic fabrics, metals, etc.) enhance realism in virtual storytelling and mixed-reality productions.

- Consistency Across Platforms
    - PBR materials ensure that assets look the same across VR headsets, AR glasses, mobile devices, and WebXR applications.

2.3 Comparative Assessment

In this section we compare the capabilities of platforms as well as asset formats regarding relevant criteria for development of XR experiences. Table 1 shows the assessment on platform level. We conclude that OpenXR is well established as an open standard for distribution of XR applications and WebXR is well-suited for web-based XR applications. Omniverse with USD as an emerging platform for professional production may be interesting for certain applications, as evaluated in more detail in section 3.

Table 1: Comparing the basic functions and features of the XR platforms.

	OpenXR	WebXR	Nvidia Omniverse	SteamVR	Oculus SDK	MRTK	AR Foundation
Open Source	✓	✓	✗	✗	✗	✓	✗
Cross Hardware / Platform Support	✓	✓	✗	✓	✗	✓	✓
Free	✓	✓	✗	✗	✗	✓	✗
Large User group	✓	✗	✗	✗	✗	✗	✗
Long Term support	✓	✗	✓	✓	✓	✗	✓
Easy to learn and adopt	✗	✓	✓	✗	✗	✗	✓
Extendable	✓	✓	✓	✗	✗	✓	✓

Table 2 shows the assessment on asset level. We conclude that glTF is the most interesting format and standard for 3D assets. However, a lot of content is still available in legacy formats and still produced in legacy formats. USD holds

promise, especially for professional production applications as it has the capability for efficient production platform integration, e.g. with Onmiverse, which is evaluated in more detail below in section 3.

Table 2: Comparing 3D asset file formats.

	Obj	FBX	glTF	USD	STL	X3D
Open Source	✓	✗	✓	✓	✓	✓
PBR texture	✗	✗	✓	✓	✗	✗
Cross Platform Support	✓	✗	✓	✗	✗	✓
Included texture and materials in the file.	✗	✓	✓	✗	✗	✓
Include shaders in file	✗	✗	✓	✓	✗	✓
3D printing format	✗	✗	✗	✗	✓	✗
View camera in file	✗	✓	✓	✓	✗	✓
Performance in VR /AR mode	✓	✓	✓	✓	✗	✓
notion of variant management / Extendable	✗	✗	✓	✓	✗	✓
Mesh compression	✗	✗	✓	✓	✗	✓
Detailed Mesh	✗	✗	✓	✓	✗	✓
Scene Light and information	✗	✗	✓	✓	✗	✓
Animation	✗	✓	✓	✓	✗	✓

3 Applications and Extensions of glTF and USD

Our evaluation of formats and standards relevant for XR applications concluded that glTF and USD are the most interesting specifications going forward. These conclusions are also widely shared in the industry, such as the Metaverse Standards Forum, which established a working group on “3D Asset Interoperability using USD and glTF”. In their Figure 3, using the analogy of images to 3D XR content, they compare glTF with JPEG as the 3D content distribution format and USD with PSD for 3D production/authoring.

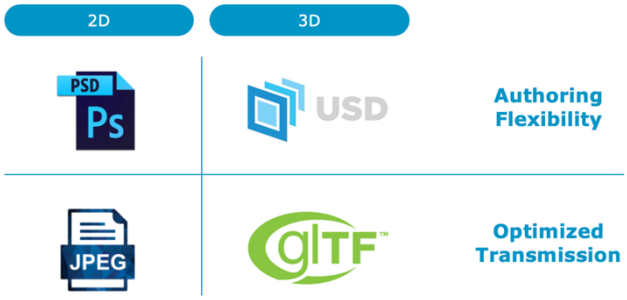

Figure 3: Analogies between 2D and 3D standards from the Metaverse Standards Forum.

When we evaluated glTF further as a 3D content standard, it was identified that volumetric video (VV) is not well supported by the standard, while VV is essential for the TRANSMIXR use cases and pilots. The reason is that the general nature of glTF follows classical computer graphics principles as a node-based specification. VV, while also being 3D geometry, also includes more video-type aspects in the temporal domain. In order to address this gap and opportunity, HSLU developed an extension for glTF for efficient support of VV, which is named VVglTF. Details about VVglTF are given in the following section 3.1.

Further, the use case Performance identified an opportunity to evaluate usage of USD in combination with Omniverse for development of the experience. The details of this evaluation are given in section 3.2.

3.1 VVglTF: An Extension to glTF for Volumetric Video

This section presents the details of HSLU's extension of glTF to support volumetric video. The work has been presented as Technical Paper at the International Broadcasting Convention (IBC) in Amsterdam in 2024. After that, we were invited to submit a corresponding paper to the SMPTE Motion Imaging Journal:

- Kit Yung Lam, Simone Croci, Philipp Haslbauer, and Aljosa Smolic, "VVGLTF: Efficient Streaming of Volumetric Video with GLTF", SMPTE Motion Imaging Journal, January/February 2025.

3.1.1 Introduction

Volumetric video (VV) is a type of video, typically created by simultaneously capturing and reconstructing 3D scenes from various perspectives, often including humans or objects, with the aim of producing an authentic and engaging experience for its viewers [6]. VV technology brings real people into virtual spaces

and facilitates 3D interaction inside the Metaverse. VV pipelines can include four major modules as shown in Figure 1. *Volumetric Capture* typically combines multiple synchronized and calibrated cameras, and potentially other sensors, often in a dedicated studio with professional lighting, green screen, etc. *Volumetric Processing* then applies advanced algorithms of computer vision and computer graphics to turn the camera signals into 3D geometry and appearance. *Volumetric Encoding* is responsible for efficient compression and streaming. Finally, *Decode and Render* cerates the output imagery and displays those to the user allowing interactive view selection. Our glTF entension named VVglTF addresses the final modules *Volumetric Encoding* and *Decode and Render*.

VV files are typically stored as dynamic 3D point clouds [1], [2] or dynamic 3D meshes [3], [4], [5] which necessitate the storage of geometric information in addition to texture (i.e. colour information). 3D point clouds specify points in 3D space with associated colour and possibly other attributes (e.g. normal, transparency), which can be rendered to create images. Similarly, 3D meshes specify 3D surface patches with associated colour and possibly other attributes (e.g. normal, transparency), which can also be rendered to create images. For time-varying data, this typically results in an enormous quantity of data that must be compressed and transmitted efficiently. However, there is no universal standard for VV file formats. Various companies and organizations have developed their own proprietary solutions. As a result, the majority of VV content is currently not accessible across several platforms to the users, while Metaverse applications require cross platform support, and users with different devices should have the same content and same user experience. To address this limitation, some initiatives have been made to establish interoperability and open specifications for VV compression, packaging, and transmission. An example of such a specification is the Volumetric Player Format Specification [23], published by the consortium Volumetric Format Association [19]. The specification outlines standardized methods for capturing and compressing extensive scenes containing more than 100 individuals, with a data range of 9 MB/s to 120 MB/s. The document further delineates techniques for establishing a linkage between a volumetric playback application and a backend server, therefore enabling streaming of volumetric data.

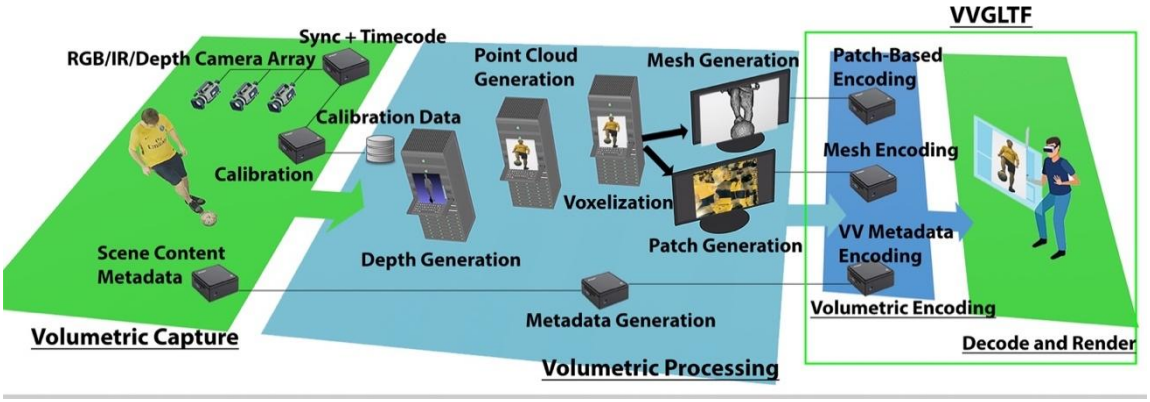

Figure 4: Typical volumetric video pipeline, VVglTF is the solution for volumetric encoding, decoding and rendering.

Another example is MPEG-5 Part 2 LCEVC [24], published by the Moving Picture Experts Group (MPEG), an international standardization organization. The specification defines a low-complexity enhancement video codec (LCEVC) that can be used to enhance the quality and efficiency of any existing or future video codec. It also supports VV coding using point clouds or meshes as input formats.

To attain the interoperability of VV data, a potential approach involves enabling VV playback using existing 3D file formats. We examined six widely used 3D file formats (OBJ, FBX, USD, glTF, X3D, STL). glTF (Graphics Library Transmission Format) emerges as the most viable standard file format when compared to the other formats (Table 3). glTF is a human-readable JSON-based (JavaScript Object Notation) 3D scene description file format, which contains a whole scene, including geometry, materials, textures, and animations [22]. It also has a binary version, which is usually referred to as GLB (GL Transmission Format Binary file). While it is not human-readable, GLB typically results in data reduction compared to text-based JSON. In our experiments below, we use GLB, although the relative savings are minor in our case, due to the large amount of geometry and texture data, which is nearly the same in both representations.

Similar to 2D video, VV has multiple frames. Each VV frame is a 3D reconstruction of the real world, and the data should be visualized frame by frame. This can be achieved via glTF extensions, which expand the functionality of the underlying glTF baseline format. Extensions provide the capability to add novel attributes, such as references to other data. Additionally, extensions can describe the structure and format of this external data. Furthermore, extensions can offer new meanings for parameters, establish reserved identification numbers, and create fresh container

formats. Extensions have the potential to be included into the core glTF specification in a subsequent version of glTF.

Table 3: Comparison of 3D file formats.

Functions / Features	Obj	FBX	glTF	USD	STL	X3D
Open source	✓	✗	✓	✓	✓	✓
PBR texture	✗	✗	✓	✓	✗	✗
Cross platform support	✓	✓	✓	✗	✗	✓
Embedded texture	✗	✗	✓	✓	✗	✗
Embedded shaders in file	✗	✗	✓	✓	✗	✗
3D printing format	✓	✗	✗	✗	✓	✗
View camera in file	✗	✓	✓	✓	✗	✓
Extendable	✗	✗	✓	✓	✗	✓
Mesh compression	✗	✗	✓	✓	✗	✗
Detailed mesh(LoD)	✗	✓	✓	✓	✗	✓
Scene light setting	✗	✓	✓	✓	✗	✓
Animation	✗	✓	✓	✓	✗	✓

To enable the VV features in glTF, we developed a custom extension following the current architecture of the glTF standard. In this specification, every frame of a video sequence has a corresponding mesh stored in our novel VVglTF file. This way the video provides a texture for every 3D mesh of the VV sequence. Using our VVglTF file format enables us to simply apply any codec with the portable file format and to support multi-platform.

The contributions of our work are as follows. First, we designed and implemented a novel cross platform glTF file format with corresponding VV player, i.e. VVglTF. Second, we developed a novel VV streaming system based on VVglTF that allows playback with adaptive streaming. This enables live streaming of VV in glTF. Finally, we conducted an experimental evaluation with VV examples, validating the efficiency of our approach.

3.1.2 Related Work

Volumetric Video Content Creation

VV (6) allows the reconstruction of real world scenes and objects in 3D, enabling visualisation and interaction with 6 DoFs (degrees of freedom), namely, interactive selection of the location and the direction of the viewpoint. This results in high interactivity and realism. In order to capture volumetric video, dedicated studios with multiple fixed cameras are typically used [3], [4]. On the other hand, capture with handheld cameras and “in-the-wild” has also been done [5]. In addition to cameras often also depth sensors are used [3].

After the acquisition with colour and optional depth cameras, usually the scene is reconstructed independently for each frame obtaining temporally inconsistent reconstructions. Then, the reconstructions are processed to get temporally consistent reconstructions [3], [4], [5]. For temporal consistency, the reconstructions are tracked obtaining registered reconstructions. In the case of meshes, the registered reconstructions have the same topology and connectivity over a certain time, which helps to define a fixed texture atlas (same resolution and UV layout from frame to frame). Temporal consistency is useful since it improves storage and streaming of the content. As outlined below, textures can be combined into videos and compressed using standard video codecs such as MP4 [26]. This will only be efficient, if textures are temporally consistent.

There are also systems, that use an image-based rendering approach instead of meshes. For example, in Project Starline [7], colour and depth images are streamed using a custom format.

Related to these VV generation methods, different storage and file formats are adopted. A universal display and data storage file format is missing, limiting interoperability and data sharing among them. VVglTF aims to provide a standardised format for VV display and data storage with our new glTF scene descriptor.

Point Cloud Compression

As an alternative to dynamic 3D meshes, VV can also be represented as dynamic 3D point clouds. Point Cloud Compression (PCC) is a method used to compress such data. MPEG-PCC [16] is an intricately developed standard for point clouds (ISO/IEC 23090-5) that includes a collection of 3D coordinate points (x, y, z) together with reflectance and RGB properties assigned to each point [17]. The MPEG-PCC standard facilitates efficient and sustainable distribution of real-world point cloud data. As an open standard, MPEG-PCC data can be easily adapted to different platforms. It could also be used in combination with VVglTF.

Point Cloud Streaming

Besides methods for compression there are methods tailored specifically for online streaming. For example, Kammerl et al. [8] encodes differences between frames computed by a double octree structure to leverage temporal redundancies. This approach does not consider user adaptation. A term used to refer to techniques

that involve user adaptation is adaptive streaming. These methods first segment the point cloud in non-overlapping regions that are encoded at different quality levels, and only the regions within the viewport are streamed at a high quality level. Examples of adaptive streaming approaches are DASH-PC [9], Park et al. [10], [11] use 3D tiles, Subramanyam et al. [12] use tiles, and Han et al. presented ViVo [13] where the point cloud is segmented into cells. Recently, machine learning models have been introduced by Huang et al. [14] and Zhang et al. [15].

3.1.3 glTF Extension Design

glTF is a standard for distribution of 3D content. It is widely adopted and enables interoperability across systems and platforms that use 3D data. The importance of glTF is increasing with growing popularity of applications such as XR (eXtended Reality) and the Metaverse. As a 3D content format, glTF follows basic principles from the computer graphics community, being a node-based scene description format. These principles are somewhat different from those in the video communication community, rather thinking in sequences of video frames. VV is both, dynamic 3D geometry with a sequence of video associated as textures. Our extension VVglTF is designed to combine the principles efficiently. A video sequence, which can be encoded in any supported standard format, is combined with a sequence of 3D meshes, which can also be encoded in any supported standard format.

VVglTF is a custom extension to enable VV playback based on the glTF 2.0 standard. It utilizes the same underlying graph structure of glTF as the core specification as depicted in Figure 5. The VV controller uses the 3D render engine, video player and glTF 3D loader API (Application Programming Interface) to control the VV playback.

VVglTF uses the video player API to decode the video, get the video duration (total number of frames), frame rate and the video frames. The VV controller converts each video frame to a texture and assigns them to the meshes. The glTF render engine only renders and shows the node elements specified in the scene description. It ignores the elements missing in the scene structure, even if the elements are embedded in the glTF file. VVglTF uses this feature to enable efficient rendering of VV with glTF. All meshes of a given sequence of VV (chunk or group-of-frames) of a certain length (number of frames, see below) are stored in the VVglTF file, but only one should be displayed at a time. This is controlled by the

video, which activates only one 3D mesh at a time and provides the corresponding texture to it.

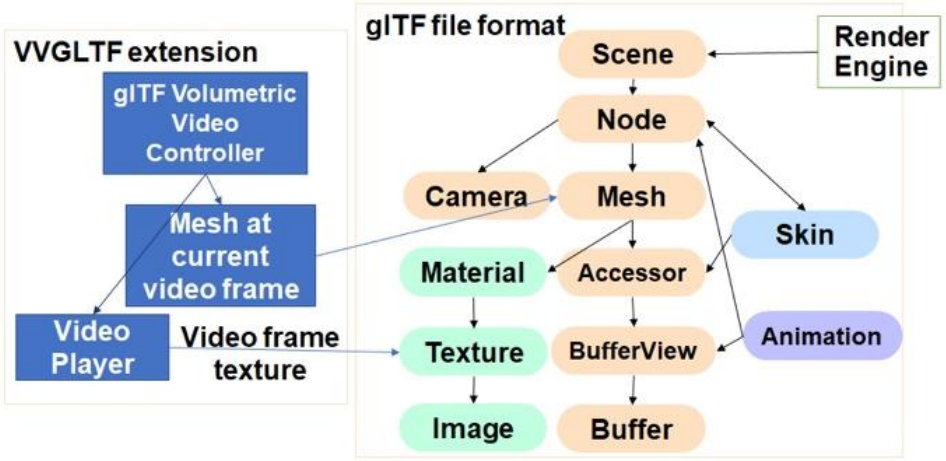

Figure 5: Design of VVglTF.

Since glTF is a scene description file, it keeps the composition of a scene containing both VV and normal 3D objects. The VV node element contains the extension properties, it keeps the video address and a list of mesh IDs corresponding to the sequence of the video. The VV controller creates the 3D object based on the node configuration and switches the mesh primitive data of the VV following the playback of the video. VVglTF keeps all 3D meshes of the VV sequence in the buffer, like any other 3D mesh in the glTF file. Each VV frame mesh has a unique mesh ID and primitive description data in the glTF file, following the glTF 2.0 specification. Any 3D mesh coding can be applied here in principle to efficiently compress the data. In our experiments below we used Draco.

The mesh primitive data in the glTF data structure can be saved either in a separate binary file or embedded in glTF JSON. The file stream loader retrieves the VV mesh data in response to sequential requests from the VV controller. A delay of VV playback may be caused by loading of mesh buffer data from the binary data. A long VV file has a longer searching time of mesh buffer data from the binary data. Proper alignment of VV frame textures with the corresponding mesh objects is crucial for smooth playback of VV. However, the loading time for video textures is often faster than the loading time for mesh primitive data due to discrepancies in resource loading time. This may result in desynchronization between the mesh and texture, as shown in Figure 6. To prevent desynchronization, the VV controller preloads multiple mesh primitive data into memory ahead of the video frame.

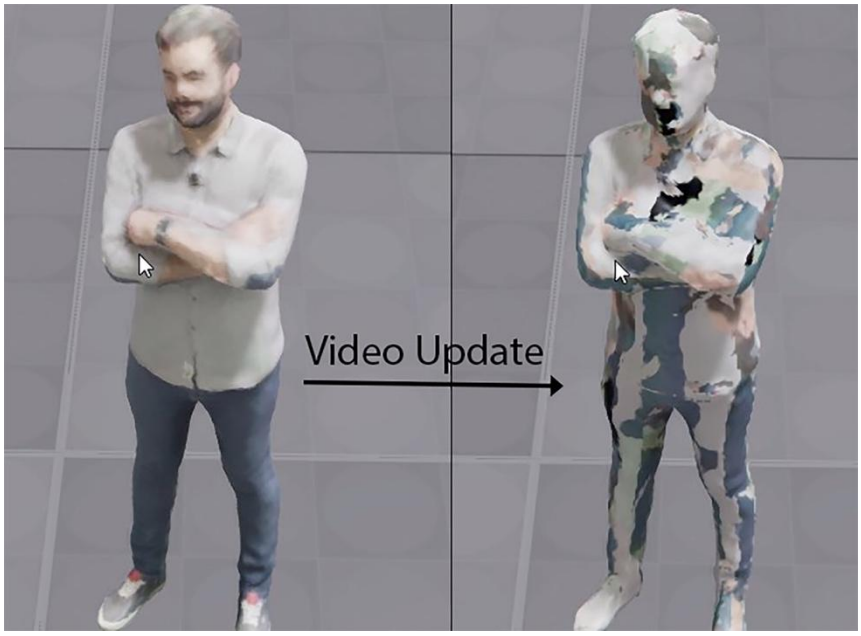

Figure 6: Desynchronisation of texture and mesh resulting in flickering.

3.1.4 Adaptive Streaming with VVglTF

We tested and applied VVglTF on different platforms including OpenXR, Unity, and WebXR. The latter for instance allows to run games and other interactive 3D content in any mobile or desktop browser. However, VV content creation can result in a tremendous amount of data (video and dynamic geometry). Long VV sequences could result in prohibitive download times. In order to solve this and to support close-to-real-time transmission, we developed an adaptive streaming system for VVglTF as illustrated in Figure 7. The VV is divided into chunks of data, each containing a certain number of frames (group-of-frames, GoF), corresponding to a certain duration. The shorter the GoF, the closer the system is to real-time transmission; however, the data overhead increases with decreasing GoF size. We evaluate this delay versus efficiency trade-off in our experiments below.

The VVglTF streaming system includes a simple server with RESTful API, VV capture clients and VV player clients (Figure 7). We support capture clients with different capturing setups, either high-end studios populated with dozens of cameras or mobile phones with mono camera. The capture client must generate 3D meshes of the capturing targets, convert the mesh textures to video, and assemble the data into VVglTF files on a GoF-by-GoF basis.

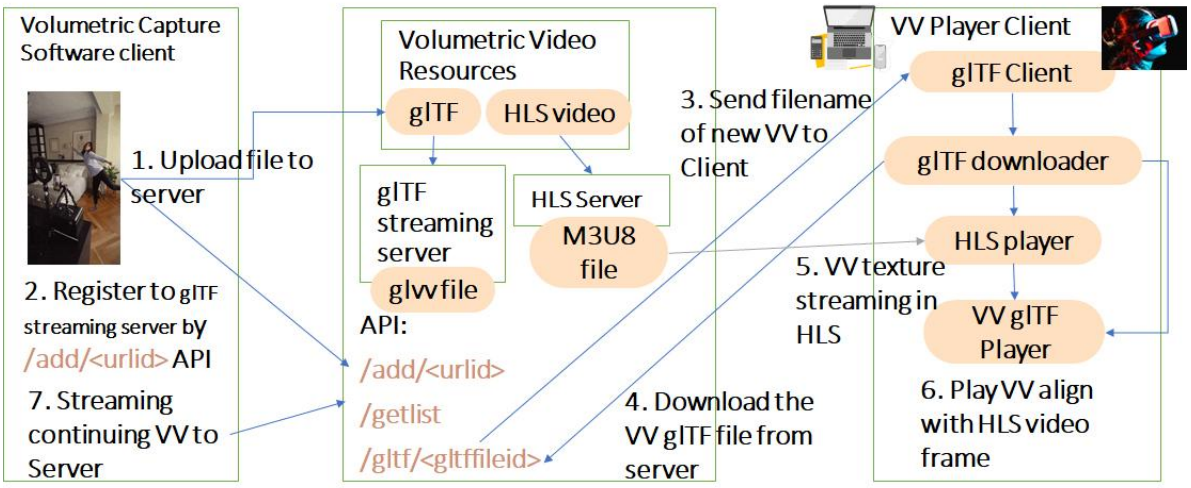

Figure 7: Architecture and data flow of our VVglTF streaming system.

The capture client uploads VV textures as video (mp4) files to the server. The server generates the HTTP Live Streaming (HLS) m3u8 files for the VV. The VV player client downloads the list of VVglTF files using an HTTP get request. If the VV is separated into multiple GoFs, the volumetric capture client provides a VV configuration file (.glvv file extension) containing the list of all VVglTF file names in the sequence of VV. The volumetric capture client continuously uploads the GoFs and updates the glvv file to the server for live streaming. At the same time, the VV player client downloads the VVglTF files and corresponding texture videos, to decode them and to visualize the content.

3.1.5 Evaluation

The GoF size is a crucial parameter in our design, as it introduces an inherent playback delay. Therefore, a smaller GoF size is desirable for delay-sensitive applications. On the other hand, it is more efficient regarding data size to pack more 3D meshes into a single GoF, as the relative overhead is reduced this way. To evaluate this trade-off, we conducted the following experiments.

Figure 8: VV examples from the vsenseVVDB2 dataset [18].

We selected 4 VV sequences (AxeGuy, LubnaFriends, Rafa2, Matis) from the vsenseVVDB2 database [18], each containing 300 VV frames at 30fps. Example views are shown in Figure 8. We encoded them into VVglTF files in binary format (GLB), first without any 3D mesh compression. We varied the GoF size as 300, 150, 60, 30, 25, 15, 5, 1, which corresponds to delays of 10s, 5s, 2s, 1s, 0.83s, 0.5s, 0.17s, and 0s, respectively. We then calculated the total file size that is needed to represent the corresponding VV sequences for each delay. The results are shown in Figure 9. We find that from a delay of 1s the data size practically saturates. Even at a delay of 0.5s the overhead is still very reasonable, a finding that is in line with common video coding standards and typical GoP sizes used there.

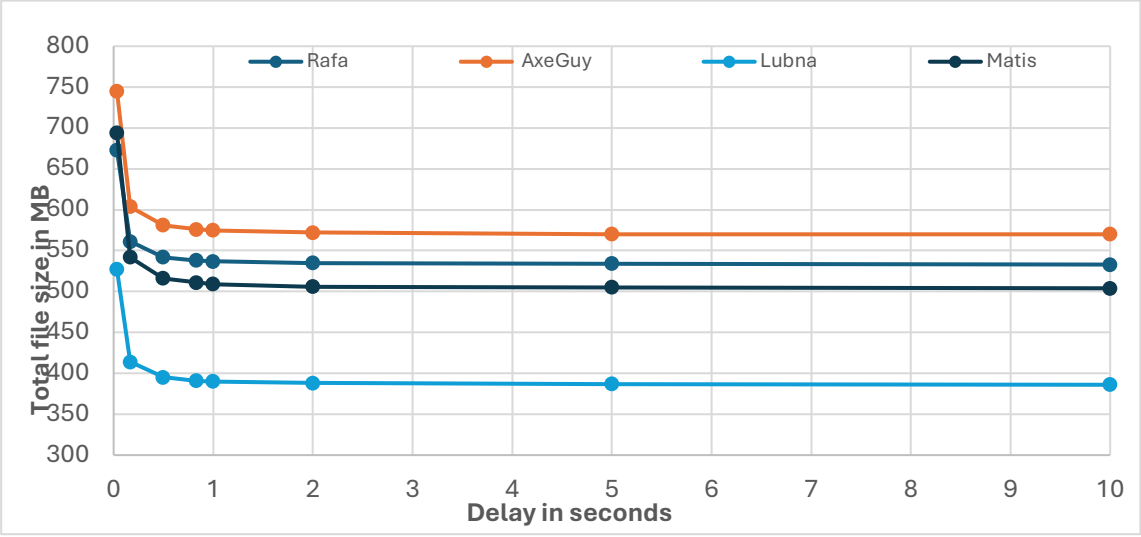

Figure 9: Total VVglTF file size (300 frames) vs. delay (based on GoF size), uncompressed.

As a second step we encoded the meshes in the VVglTF files using the commonly used 3D mesh codec Draco [25], which is available as standard in glTF. For that we used typical settings of Draco (level 5) providing good visual quality (i.e. almost no coding artifacts visible). The results are shown in Figure 10 (please note different scaling of the x-axis). Again, we find that reasonable delays of 0.5s or less are possible without substantial increase of the data size. We further notice that the overall data rate is decreased by a factor of 10 compared to the uncompressed version. The additional data for the separate video files at good visual quality are: Rafa 44.6MB, AxeGuy 45MB, Lubna 48.8MB, Matis 49.9MB. Overall, we therefore get a reduction from approximately 600-800MB raw data rate to 100-150MB compressed data rate at a good viewing experience. This validates that VVglTF is well suited for efficient representation of VV.

VVglTF has been used in several follow-up projects, where characters encoded in VVglTF have been integrated into full VR experiences via game engines Unity and Unreal. An application for an enhanced AR museum experience has been presented and evaluated in [20]. Additional tests have been made with VVglTF content in full WebXR scenes.

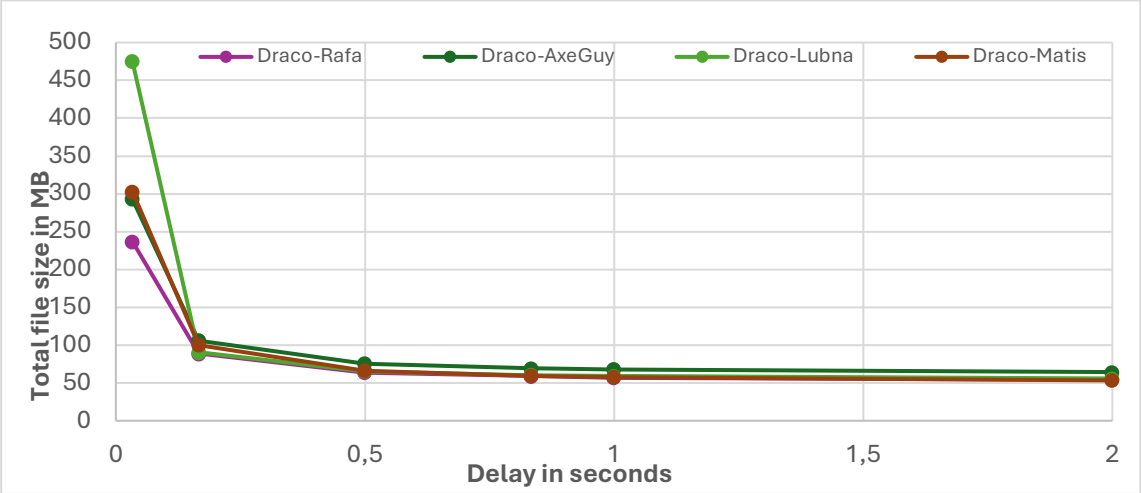

Figure 10: Total VVglTF file size (300 frames) vs. delay (based on GoF size), Draco compressed.

3.1.6 Conclusions

This report presents VVglTF and serves as a groundwork for enabling VV in glTF, which is an important standard for 3D content, ensuring interoperability and efficiency across platforms and systems. We tested and applied it via OpenXR, Unity, Unreal, and WebXR. With VVglTF it is possible to bring VV for example into

mobile and desktop browsers and to use it in corresponding 3D applications. For efficient streaming, we developed an adaptive approach, including standard video streaming components such as HLS. Our experiments validate that VVglTF is efficient regarding delay and compression.

This evaluation does not consider limitations arising from terminal devices such as mobiles or head mounted displays, which may cause additional delays or performance problems. In our future work, we will consider such difficulties on low performance devices (in particular standalone XR headsets). We will further investigate live streaming for real-time holographic communication, and combination with other types of VV representations such as point clouds or Gaussian splatting.

VVglTF has been made available to the public on the github [21] of the Khronos Group [22], which standardizes glTF.

3.2 USD

The Use Case Performance led by SATORE is essentially a distributed and collaborative XR application production. Various types of 3D assets including VV and motion capture are combined into an immersive theatre experience. This scenario is exactly what the open framework of USD is designed for. The partners therefore decided to practically evaluate the use of USD for the production. SATORE was able to organize free access for all involved partners to NVIDIA's USD platform Omniverse. The partners including SATORE, TUS, CWI, Intel, TCD, Khora, and HSLU then tested and evaluated the capabilities regarding several parameters, including support for 3D assets, support for audio, support for VV, compatibility with game engines, system performance, and (real-time) collaboration. It was found that the Omniverse platform offers significant potential for remote collaboration in 3D production, while not all features are sufficiently developed yet. This includes scripts and connectors and efficient support for VV, similarly to glTF as outlined above.

In order to address this gap and opportunity as identified, HSLU developed efficient support of VV in USD/Omniverse. The solution is similar to the glTF extension shown above. An example of VV in USD/Omniverse is illustrated in Figure 11.

However, in-depth assessment revealed that not all tools needed for the production are available in Omniverse, which is not a game engine. Some plugins and drivers,

which are essential for the production and available in Unreal, are not sufficiently supported in Omniverse yet. We concluded that Omniverse is still in development, not mature enough yet, and therefore decided not to base the production on this platform but rather to stay in Unreal. There is still the option to port the full final experience to Omniverse. It was also useful during production for checks of USD content, whether it executes the same in Unreal and Omniverse.

USD itself was found to be very useful on asset level, for instance for motion capture data. Also, many 3D elements were produced in Houdini and exported as USD to Unreal.

Figure 11: Examples of volumetric video in Omniverse.

4 Usage of Formats and Standards in TRANSMIXR Use Cases

The TRANSMIXR use cases started designing and implementing different XR experiences, taking the initial conclusions and recommendations on formats and standards into account. An overview of the finally applied formats and standards per TRANSMIXR experience is shown in Table 4.

Three of these are VR experiences and based their development on Unity. The fourth, Performance is a see-trough MR experience and used Unreal instead. This

makes sense, as Unity is well-established for development of VR, while Unreal is standard in professional production which the use case Performance relates to.

All experiences use Quest 3 as their primary HW platform, some in standalone mode (Broadcast, Performance), some in tethered mode (Planica, Museum) with a PC/laptop due to computational complexity.

All experiences finally use OpenXR as their general format on platform level, which aligns to the initial assessment. OpenXR is well-suited for the requirements of all use cases and can therefore be considered TRANSMIXR's "eXtended Reality Media Experience Format".

While Planica is an offline single user experience, the other 3 experiences can include multiple users and real-time elements.

At asset level, the experiences use a variety of different 2D and 3D media in related formats, which results in a rich and insightful overall picture. Volumetric video is used in all experiences. Planica and Performance use offline VV in our novel VVglTF format. Broadcast and Museum apply real-time VV streaming in point cloud format, based on a pre-existing system from the VR2Gather project provided by CWI.

For 2D video standard formats like MP4 are used. Performance applies BINK (by EPIC), which is widely used in the games world and the Unreal eco-system. Broadcast applies real-time video streaming in RTSP for 2D and also 360 video.

Images are included in standard formats like JPEG and PNG.

For 3D objects, animations and mocap data mostly FBX was used, which is most established in this area. Performance also widely adopted USD. glTF may gain more traction in this space in the future.

Avatars were applied in Performance and Museum. Both used internal formats of the corresponding game engines Unreal and Unity.

Audio was included in all experiences in standard formats such as AAC, WAV, MP3, MP4.

Performance applied additional formats for communication and AI components.

Planica applied the novel content type Gaussian Splatting (GS), which is typically exported from content creation tools in a dedicated PLY format, which is

uncompressed. The Unity plugin that was used for the experience applies a compressed GS format to increase efficiency.

However, as 3D Gaussian Splatting enjoys increasing popularity and dissemination in end user applications, standardization has become a very active topic in committees such as MPEG/JPEG [28].

There is a draft for the glTF standard to extend the format with Gaussian Splatting (KHR_gaussian_splatting) which is missing the spherical harmonics as the draft authors intend to handle them in separate extensions [29].

Niantic, the company responsible for Pokémon Go, implemented their own open-source compressed splat format ".spz". Combining quantization, scaling, and zip compression, the format reaches a file size reduction of 90% as compared to uncompressed .ply files [27].

The web-based 3DGS renderer gsplat.js implements its own compact splat format, with an apt file extension named ".splat". This format is more compact because it lacks the spherical harmonics responsible for view-dependent appearance [29].

Nerfstudio (or gsplat) implements compression for splat .ply files that uses PNG compression and quantization to reduce the size of splats to a mere 8MB per 500K Gaussians [30] which is a size reduction of more than 90%. The implementation builds on the paper that introduced Self-Organizing Gaussian Grids [32].

Supersplat also implements a compression scheme for .ply files which results in more than 90% size reduction. The implementation is based on insights from the person who wrote the Unity plugin that we used [33].

We conclude that compression of GS data holds potential for future research.

Table 4: Application of formats and standards in TRANSMIXR XR experiences.

	XR Experience			
	Planica	**Broadcast**	**Performance**	**Museum**
Type	VR	VR	MR	VR
SW Platform	Unity	Unity	Unreal	Unity
HW Platform	Quest 3	Quest 3	Quest 3	PC VR - Meta Link (for Quest 3 and 2, Quest Pro)
Experience format	OpenXR Runtime (tested with SteamVR & Quest Link)	OpenXR (Tested with Android Standalone)	OpenXR + Meta extensions	Open XR
Real-time media	no	yes	yes	yes (via WebRTC, DASH)
Single/multi user	single	multi	single + multi	multi
Assets				
Volumetric video	VVglTF	Point Cloud (cwipc, MPEG Anchor Codec)	VVglTF	Point Cloud (cwipc, MPEG Anchor Codec)
2D video	MP4	RTSP (real-time streaming)	BINK	MP4
360 video		RTSP (real-time streaming)		
Images	PNG	PNG	JPEG	JPEG
3D objects	FBX	FBX	USD, FBX	FBX
Audio	AAC/MP4	PCM (24bit @ 48kHz), AAC/MP4	AAC/MP4	AAC/MP4
Avatars			Metahuman, Unreal rigged skeleton model	Unity standard rigged skeleton model
Gaussian Splats	PLY, .asset (Compressed intermediate file format by Unity Gaussian Splatting plugin)			

5 Summary and Conclusions

Interoperability of systems is crucial for creative and cultural industries. However, in the area of extended (XR) realities, formats and standards are not yet as mature and established as for standard media like video, images, or audio. Therefore, the main objective of T3.1 was to ensure that TRANSMIXR use cases and technologies are sufficiently supported in terms of formats and standards.

Work in T3.1 started early in the project with an initial overview and assessment of XR-related formats and standards. This gave initial recommendations for design and development. At a platform level, OpenXR was identified as the most promising format for general XR experiences. For web applications, WebXR was recommended. At asset level, it was found that many types of content are already well-supported, however this space seems quite fragmented with several specifications for similar purposes. USD and glTF were identified as promising developments going forward, and this is also reflected in industry activities.

A need and opportunity was identified with as volumetric video not yet supported in glTF. To address this, the project developed an extension to glTF named VVglTF. This was published and submitted to a standards organization (Khronos). It was also used in TRANSMIXR's experiences Planica and Performance, and in a follow up project of HSLU. USD was evaluated with the nVidia Omniverse platform in the Performance use case. While USD proved very useful at an asset level, Omniverse is not yet mature enough on platform level.

A variety of XR experiences were developed in TRANSMIXR, including both VR and MR. All of them used OpenXR as their platform format, following the initial assessment and establishing it as TRANSMIXR's "eXtended Reality Media Experience Format". Meta Quest 3, in standalone and tethered modes, was selected as the main hardware platform. Unity and Unreal were applied as game engines.

At an asset level, the experiences use a variety of different 2D and 3D media in related formats, which results in a rich and insightful overall picture. Volumetric video is used in all experiences. VVglTF is used for offline cases, while real-time streaming is done using the VR2Gather system. Standardized real-time streaming of volumetric video is an area of opportunity for further development.

Other, more standard media (3D objects, animations, mocap, avatars, 2D video, 360 video, images, audio, etc.) are well supported by existing formats and standards, but fragmentation is evident in this context. USD and glTF have the potential to streamline and unify this space in the future.

Gaussian Splatting, as a novel type of 3D asset, has a lot of potential for future work, including in the context of formats and standards. For instance, efficient compression is an active area of research.

Overall, we conclude that the objectives related to XR formats and standards could be achieved and that generally the field is maturing, while challenges and opportunities remain.

6 References

[1] Reimat, I., Alexiou, E., Jansen, J., Viola, I., Subramanyam, S., & Cesar, P., 2021. Cwipc-sxr: Point cloud dynamic human dataset for social xr. Proceedings of the 12th ACM Multimedia Systems Conference. pp. 300 to 306.

[2] Sterzentsenko, V., Karakottas, A., Papachristou, A., Zioulis, N., Doumanoglou, A., Zarpalas, D., & Daras, P., 2018. A low-cost, flexible and portable volumetric capturing system. 2018 14th international conference on signal-image technology & internet-based systems, pp. 200-207.

[3] Collet, A., Chuang, M., Sweeney, P., Gillett, D., Evseev, D., Calabrese, D., ... & Sullivan, S., 2015. High-quality streamable free-viewpoint video. ACM Transactions on Graphics (ToG).

[4] Hilsmann, A., Fechteler, P., Morgenstern, W., Paier, W., Feldmann, I., Schreer, O., & Eisert, P., 2020. Going beyond free viewpoint: creating animatable volumetric video of human performances. IET Computer Vision, 14(6), pp. 350 to 358.

[5] Pagés, R., Amplianitis, K., Monaghan, D., Ondřej, J., & Smolić, A., 2018. Affordable content creation for free-viewpoint video and VR/AR applications. Journal of Visual Communication and Image Representation, 53, pp. 192 to 201.

[6] Valenzise, G., Alain, M., Zerman, E., & Ozcinar, C. (Eds.)., 2022. Immersive Video Technologies.

[7] Lawrence, J., Goldman, D., Achar, S., Blascovich, G. M., Desloge, J. G., Fortes, T., ... & Tong, K., 2021. Project starline: A high-fidelity telepresence system. ACM Transactions on Graphics (TOG), 40(6), pp. 1 to 16.

[8] Kammerl, J., Blodow, N., Rusu, R. B., Gedikli, S., Beetz, M., & Steinbach, E., 2012, May. Real-time compression of point cloud streams. 2012 IEEE ICRA, pp. 778 to 785.

[9] Hosseini, M., & Timmerer, C., 2018, June. Dynamic adaptive point cloud streaming. Proceedings of the 23rd Packet Video Workshop, pp. 25 to 30.

[10] Park, J., Chou, P. A., & Hwang, J. N., 2019. Rate-utility optimized streaming of volumetric media for augmented reality. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 9(1), pp.149 to 162.

[11] Park, J., Chou, P. A., & Hwang, J. N., 2018, December. Volumetric media streaming for augmented reality. 2018 IEEE GLOBECOM, pp. 1 to 6.

[12] Subramanyam, S., Viola, I., Hanjalic, A., & Cesar, P., 2020, October. User centered adaptive streaming of dynamic point clouds with low complexity tiling. Proceedings of the 28th ACM international conference on multimedia, pp. 3669 to 3677.

[13] Han, B., Liu, Y., & Qian, F., 2020, April. ViVo: Visibility-aware mobile volumetric video streaming. Proceedings of the 26th annual international conference on mobile computing and networking, pp. 1 to 13.

[14] Huang, Y., Zhu, Y., Qiao, X., Tan, Z., & Bai, B. (2021, October). Aitransfer: Progressive ai-powered transmission for real-time point cloud video streaming. Proceedings of the 29th ACMMM, pp. 3989 to 3997.

[15] Zhang, A., Wang, C., Han, B., & Qian, F., 2021, February. Efficient volumetric video streaming through super resolution. Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications, pp. 106 to 111.

[16] Schwarz, S., Preda, M., Baroncini, V., Budagavi, M., Cesar, P., Chou, P. A., ... & Zakharchenko, V. (2018). Emerging MPEG standards for point cloud compression. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 9(1), 133-148.

[17] Ilola, L., Kondrad, L., Schwarz, S., & Hamza, A., 2022. An overview of the mpeg standard for storage and transport of visual volumetric video-based coding. Frontiers in Signal Processing, 2, 883943.

[18] Zerman, E., Ozcinar, C., Gao, P., & Smolic, A., 2020, May. Textured mesh vs coloured point cloud: A subjective study for volumetric video compression. Twelfth QoMEX, pp. 1 to 6).

[19] https://www.volumetricformat.org

[20] Anh Nguyen, Jan Lässig, Anna F. Kälin, Ariana Huwiler, Philipp Haslbauer, Gareth W. Young, Lam Kit Yung, Aljosa Smolic, “A Volumetric Video

Application to Enhance Museum Experiences", VRST '24: Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology. https://dl.acm.org/doi/10.1145/3641825.3689695

[21] https://github.com/KhronosGroup/glTF/pull/2438

[22] https://www.khronos.org/api/index_2017/gltf

[23] https://www.volumetricformat.org/specifications

[24] https://www.lcevc.org

[25] https://github.com/google/draco

[26] Information technology – Coding of audio-visual objects – Part 14: MP4 file format (Standard) (3rd ed.). ISO. January 2020. ISO/IEC 14496-14:2020.

[27] B. Huang, Z. Yu, A. Chen, A. Geiger and S. Gao, "2d gaussian splatting for geometrically accurate radiance fields," in ACM SIGGRAPH 2024 conference papers, 2024.

[28] "Joint JPEG MPEG Workshop on Radiance Fields," 1 January 2025. [Online]. Available: https://www.mpeg.org/joint-jpeg-mpeg-workshop-on-radiance-fields/.

[29] "KHR_gaussian_splatting," [Online]. Available: https://github.com/CesiumGS/glTF/tree/proposal-KHR_gaussian_splatting/extensions/2.0/Khronos/KHR_gaussian_splatting. [Accessed 4 March 2025].

[30] dylanebert, "gsplat.js README," [Online]. Available: https://github.com/huggingface/gsplat.js/blob/main/README.md. [Accessed 3 March 2025].

[31] "gsplat Compression," [Online]. Available: https://docs.gsplat.studio/main/apis/compression.html. [Accessed 3 March 2025].

[32] W. Morgenstern, F. Barthel, A. Hilsmann and P. Eisert, "Compact 3d scene representation via self-organizing gaussian grids," in European Conference on Computer Vision, 2024.

[33] "Compressing Gaussian Splats," [Online]. Available: https://blog.playcanvas.com/compressing-gaussian-splats/. [Accessed 4 March 2025].