This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101092889, Topic HORIZON-CL4-2022-HUMAN-01-14

SHARESPACE

Embodied Social Experiences in Hybrid Shared Spaces

Project Reference No	101092889
Deliverable	D5.6. XR communication platform design and validation v1
Workpackage	WP5: Overall Framework Definition
Nature	DEM (Demonstrator)
Dissemination Level	PU - Public
Date	31/05/2024
Status	v1.0
Editor(s)	Emmanuel, Helbert (ALE) Daniel Rammer (ARS) Stephane Donikian (GOLAEM)
Involved Institutions	ALE, DFKI, GOLAEM, CYEN, ARS
Document Description	The deliverable presents the XR-communication platform which will be used for the trials

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Table 1: List of Abbreviations

Term / Abbreviation	Definition
AI	Artificial Intelligence
MSI	Multi-Sphere Image
VR	Virtual Reality
XR	eXtended Reality
HMD	Head Mounted Device

1 INTRODUCTION

This deliverable presents the design, the integration and test and validation of the XR collaboration platform. The XR collaboration platform is tightly linked with the motion-capture components as well as the rendering components. It shall deliver real-time communication means to distribute audio, kinematics, and gaze information from multiple sources to multiple endpoints where avatars animation will be performed. This document will describe in detail the various components which build the XR communication platform and how they interact. It will also depict the iterative method the team followed to integrate and validate these components in a functional step-by-step approach. Finally, it will provide recording of integration tests to demonstrate the functional achievements of the XR collaboration platform and assess its performance required for the given scenarios.

The demonstrations showcase real-time animation of several avatars based on motion captured remotely. They include not only skeleton animation but also face animation based on audio.

This deliverable corresponds to the first milestone where the main goal was to reach an end-to-end functional platform. The second step shall provide a higher integration degree with better performance and scalability and may also include face animation with eye-tracking.

The Figure 1 below describes the general test set-up used to validate the good functioning of the platform. For testing purposes, body kinematics is captured with a mocap suit. Audio can be gathered either from the default microphone of the computer connected with the XR communication platform or from the HMD mic. The animation and rendering, as well as the audio and lips animation are checked at the receiving side on a computer screen or on the HMD. On the diagram, StreamId_x represents the couple of streams made of audio and encoded kinematics. In the configuration presented in the diagram, each rendering application (on the right) will then receive four different streams, one couple of audio and kinematics from each emitter.

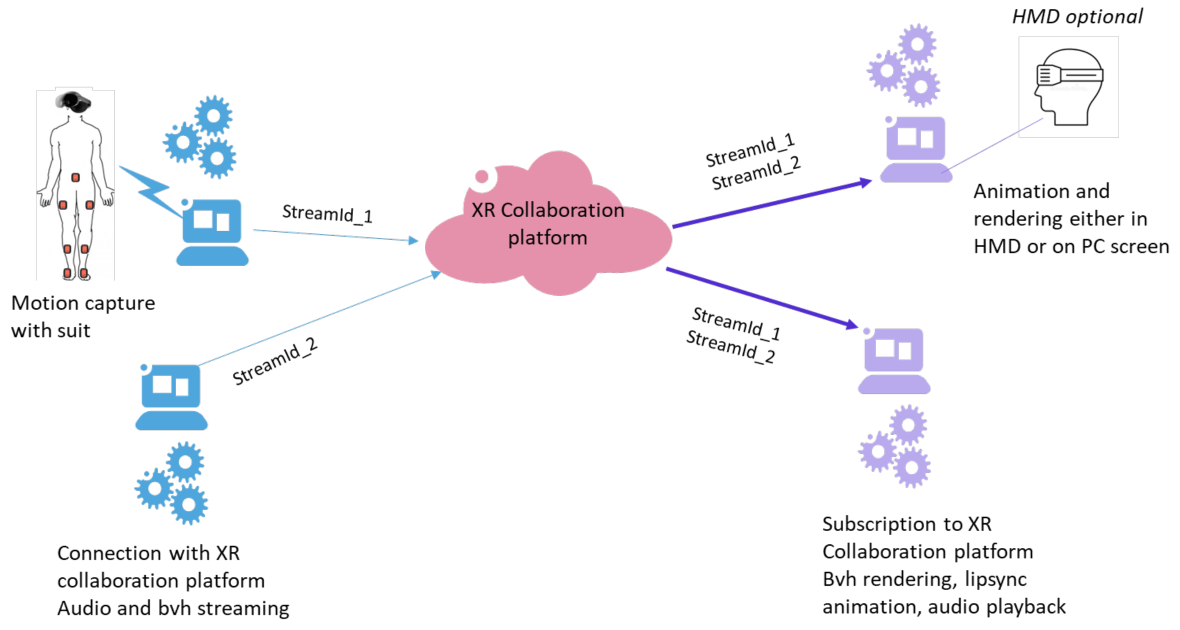

HMD optional
StreamId_1
StreamId_2
Animation and rendering either in HMD or on PC screen
StreamId_1
XR Collaboration platform
Motion capture with suit
StreamId_1
StreamId_2
StreamId_2
Connection with XR collaboration platform Audio and bvh streaming
Subscription to XR Collaboration platform Bvh rendering, lipsync animation, audio playback

Figure 1 – General test architecture for demonstrators

2 BACKGROUND

The purpose of the XR Communication Platform is to provide an integrated collaboration infrastructure to connect users in VR or AR mode. In VR mode, the platform will support the transmission of data representing in real-time the kinematics of a user. This kinematic information, encoded into a .bvh format is streamed by the infrastructure to the viewers wearing HMD where the L1 or L2 avatar of the user is animated. In addition to the kinematics, the infrastructure also transmits and distributes the audio gathered at the HMD microphone. This XR Communication Platform is one of the key Sharespace component involved in the “Sharespace for Health” real world scenario. In phase one of this scenario (See Deliverable D1.2[1], §2.2.1), two patients will be wearing HMD where they will be able to see the real-time animated avatar of the therapist. Hence, the XR Communication platform shall ensure the communication set-up between 3 HMD and the transmission of audio and encoded kinematics of the therapist towards the two HMD worn by the two patients with as low delay as possible. This phase one “evaluation” will deliver the results of a usability study.

3 APPROACH

3.1 ARCHITECTURE REMINDER

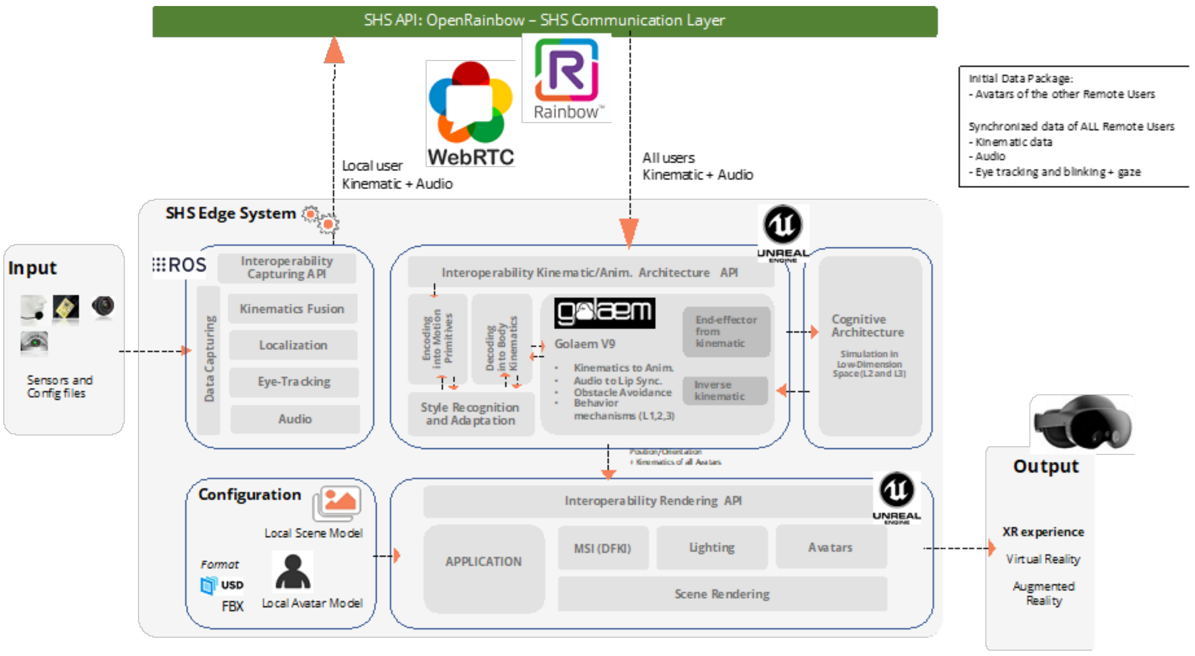

SHS API: OpenRainbow – SHS Communication Layer
WebRTC
Rainbow
Local user
Kinematic + Audio
All users
Kinematic + Audio
Initial Data Package:
- Avatars of the other Remote Users
Synchronized data of ALL Remote Users
- Kinematic data
- Audio
- Eye tracking and blinking + gaze
SHS Edge System
Input
Sensors and Config files
ROS
Interoperability Capturing API
Data Capturing
Kinematics Fusion
Localization
Eye-Tracking
Audio
Interoperability Kinematic/Anim. Architecture API
Encoding into Motion Primitives
Decoding into Body Kinematics
Style Recognition and Adaptation
golaem
Golaem V9
Kinematics to Anim.
Audio to Lip Sync.
Obstacle Avoidance Behavior mechanisms (L1,2,3)
End-effector from kinematic
Inverse kinematic
UNREAL ENGINE
Cognitive Architecture
Simulation in Low-Dimension Space (L2 and L3)
Position/Orientation + Kinematics of all Avatars
Configuration
Local Scene Model
Format
USD
FBX
Local Avatar Model
Interoperability Rendering API
UNREAL ENGINE
APPLICATION
MSI (DFKI)
Lighting
Avatars
Scene Rendering
Output
XR experience
Virtual Reality
Augmented Reality

Figure 2 – SHS Architecture

As far as the XR communication platform is concerned, we can split the system in three main blocks. The first block is in charge of capturing a human behaviour (L0) at one side, meaning, kinematics, voice or sound produced, face and eyes movements. All this information must be encoded and prepared for streaming towards remote recipients. This is represented at the top left side of Figure 2.

[1] Sharespace Deliverable D1.2: Research Requirements for challenges and scenarios v1. 2022

The second block, at the receiving side (top middle block of Figure 2), is in charge to decode the information, synchronize, if necessary, the heterogeneous data received in separate streams, and animate the corresponding avatar (L1) through a rendering device. The avatar animation may be modified thanks to the cognitive architecture component (top right block in Figure 2) to insert amplification or attenuation into avatar kinematics (L2). The third block (SHS communication layer in Figure 2), interconnected between the two former ones, is in charge of ensuring an end-to-end communication channel to support the multiple streams sent from the emitters (L0) and the receivers (L1 or L2). In addition, it shall support routing to ensure connection between multi-users. An emitter may stream all its encoded information towards several recipients. The user device of each recipient will render the same L1 or L2 avatar. The same, a recipient may receive streams from several users (L0) and shall render on the same user device several L1 or L2 avatars.

3.1.1 Emitter

the emitter, developed by DFKI, is responsible for encoding the kinematics of the user, gathering the audio from the HMD and possibly data from the eye-tracking system integrated in the HMD (see D3.7[2]). Though Sharespace ambition is to provide a solution to collect kinematics with as low sensors as possible, including the use of HMD for the kinematics of the upper body (see D3.1[3] and D3.3[4]), we decided to use a mocap suit with integrated sensors to simplify the test set-up. The emitter also manages the signaling with the infrastructure to indicate the availability of data stream as described in Figure 4. As soon as a renderer willing to consume this data connect to the infrastructure, a WebRTC data channel is established between the emitter and the renderer, and the data issued by the HMD and encoded by the emitter are streamed directly to the renderer.

3.1.2 Renderer

The renderer is in charge to connect to the infrastructure, initiate direct connection with selected emitters based on id of streams, consume and decode audio and kinematics contained in bvh streams and audio streams. The main role of the renderer is to animate avatars in a VR environment displayed in an HMD (See deliverable D4.1[5]). Each avatar is associated to a specific user and identified with the stream id of the corresponding emitter. Information in bvh is used to animate the skeleton of the avatar. Audio is injected into Unreal Engine plugin and playback in HMD speakers. The animation of the lips of the avatar is based on the speech analysis in audio (See deliverable D4.3[6]). Only English language is supported at this stage.

3.1.3 Communication infrastructure

The communication infrastructure is built around the component Pixel Streaming [7] server, hosted in the Rainbow cloud infrastructure, and a Pixel Streaming plugin running at the receiver side. This plugin ensures a WebRTC data connection with the Pixel Streaming server. It subscribes at the Pixel Streaming server to selected Stream ids related to the streams sent by the emitter. At the emitter side, a transmitting component connect with the Pixel Streaming server with a specific Stream Id that will be used to route the corresponding streams to the renderers which have subscribed to it.

[2] Sharespace Deliverable D3.8: Multi-sensors and Multifocal XR display v1
[3] Sharespace Deliverable D3.1: Self-calibrating ego centric visual-inertial body tracking v1. 2024
[4] Sharespace Deliverable D3.3: Accurate Hand Kinematics Predictions from ego-vision v1. 2024
[5] Sharespace Deliverable D4.1: Definition of SHARESPACE Avatar. December 2023
[6] Sharespace Deliverable D4.3: Avatar animation v1. 2024
[7] https://dev.epicgames.com/documentation/en-us/unreal-engine/pixel-streaming-in-unreal-engine?application_version=5.0

The sequence diagram in Figure 4 describes how the infrastructure manage the connection set-up between the emitters (kinematics and audio) at one side, and the renderers (avatars animation with audio and lipsync) at the other side. When an emitter is ready to stream encoded kinematics, audio or other data, it indicates the id of its stream to the infrastructure. At the opposite, when a renderer is ready to consume and decode kinematics, audio or other data, it polls regularly the infrastructure for the selected streams to connect. As soon as an awaited stream is available, the renderer initiates a data channel negotiation where the emitter can stream all the data (encoded kinematics, audio or other) associated to the stream id of the emitter.

When the emitter ends the connection, the data channel is closed. All renderers having subscribed to this stream id are notified and then, restart the polling for the stream to reconnect.

3.1.4 XR Collaboration platform for Ars Electronica infrastructure

In addition to the communication infrastructure described above, Unreal Multiplayer Replication [8]is also utilized in the Shared Creativity scenario. This enables the synchronization of arbitrary entities, such as moving or falling objects, within the application. Therefor the users, remote users and the instances running on the Deep Space 8K machines are connected to a game server. The game server can either run within the Ars Electronica network or globally available on a cloud server. As Unreal's Replication system is regularly used in the games industry, one can generally assume a certain degree of robustness. However, a setup and tests with a server running within the Ars Electronica's network, two instances running on the Deep Space 8K machines and one instance running on an additional PC have proven the feasibility of the utilization of this technology for the Shared Creativity scenario performances.

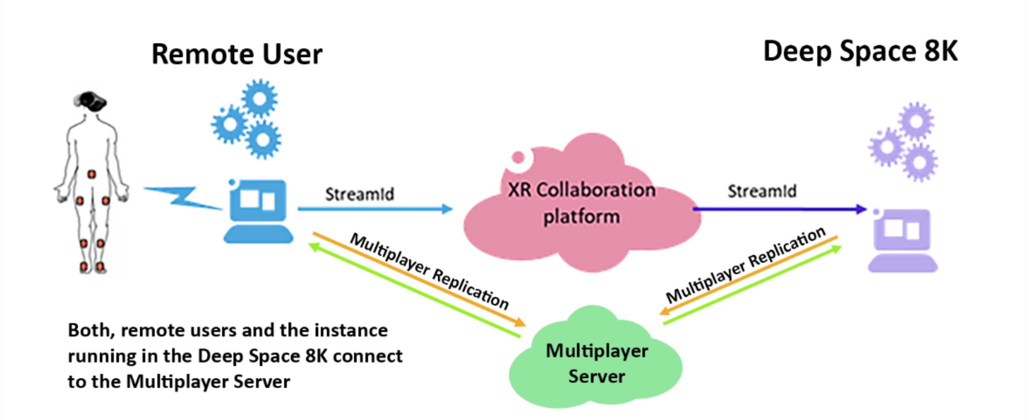

Remote User
Deep Space 8K
StreamId
XR Collaboration platform
StreamId
Multiplayer Replication
Multiplayer Replication
Both, remote users and the instance running in the Deep Space 8K connect to the Multiplayer Server
Multiplayer Server

Figure 3 - XR Collaboration platform with Multiplayer server

[8] https://dev.epicgames.com/documentation/en-us/unreal-engine/networking-and-multiplayer-in-unreal-engine?application_version=5.3

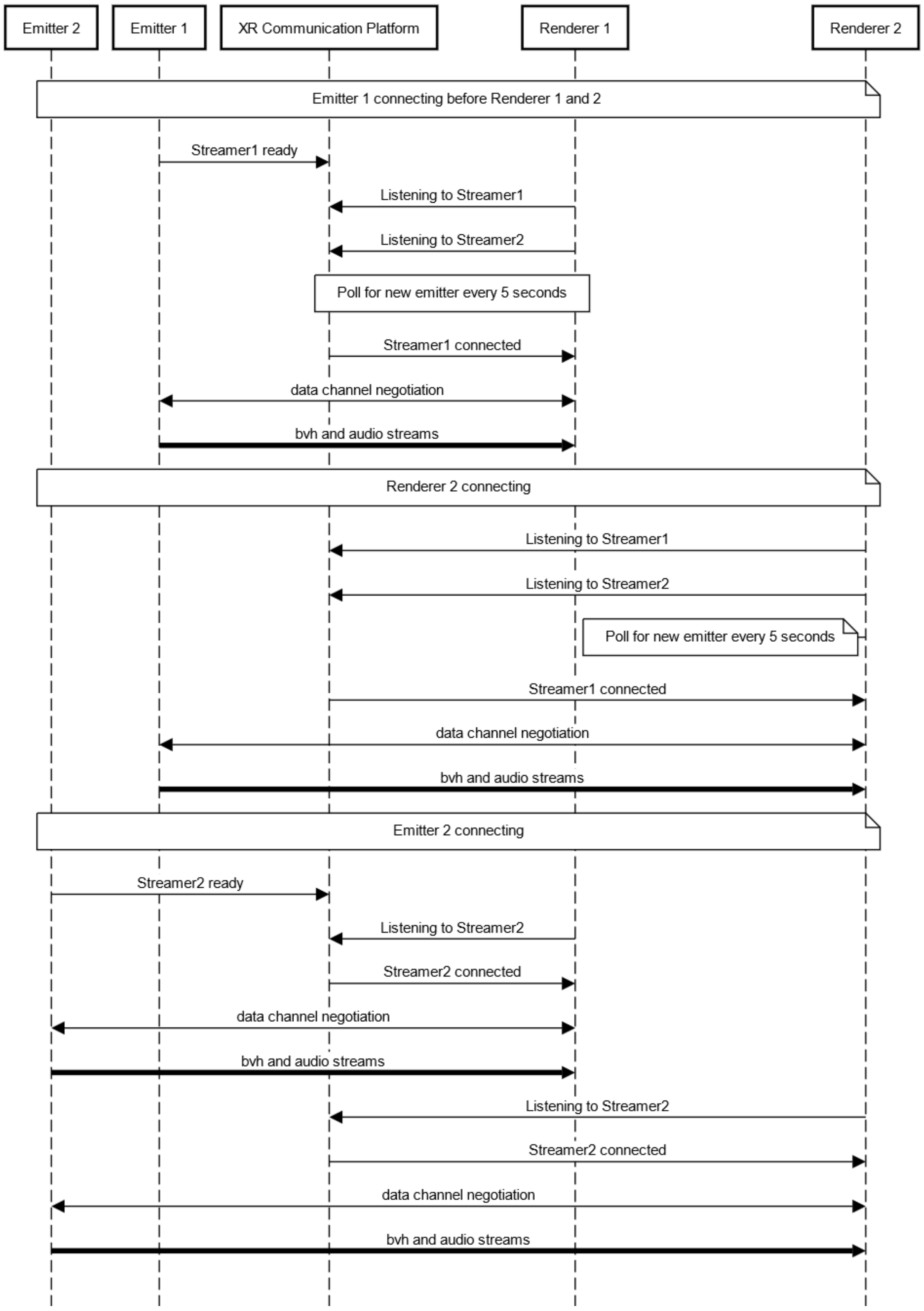


Figure 4 - Sequence diagram for multi-users connection set-up

3.2 Integration

The design, development, integration and test follow an iterative methodology where the objective is to add one service after the other. These iterations aim at validating each functional block while avoiding as much as possible dependencies among the different services supported by the platform. The table x below summarizes the various iterations and their purposes.

Id	Summary	Purpose
1	Kinematics encoding and decoding	Test accuracy of kinematics encoding/decoding
2	Renderer connection with XR collaboration platform	Test WebRTC data channel with renderer
3	Full transmission chain	Test WebRTC data channel for audio and bvh between emitter and renderer
4	Multi-users, emission and rendering	Test ability of XR collaboration architecture to route multiple streams and renderers to animate several avatars
5	Lipsync	Test lip animation based on audio and synchronization of face animation with body animation
6	Audio+Capturing/Rendering on same HMD	Test the overall performances of the platform in a pilot scenario: multi-users, motion and audio capture and rendering on the same HMD

Table 2 - Test Plan

We depict in detail each test below:

- **Iteration 1**: Test of kinematics encoding and decoding. At the emitter side, the objective is to encode into .bvh files the kinematics of a human body while using a combination of camera and sensors. These files can then be used at the rendering side for both validating the decoding of the animation chain with a reference .bvh file and validating the motion capture precision and accuracy.

- **Iteration 2**: Test of the connection between the renderer and the Pixelstreaming server and animation of an avatar from a .bvh file stored remotely. This step enables validating the full downstream chain without relying on the emitter side. This test is made possible thanks to a simulation tool which mimics an emitter and supports the streaming of stored .bvh file as well as audio streams from local audio capturing devices.

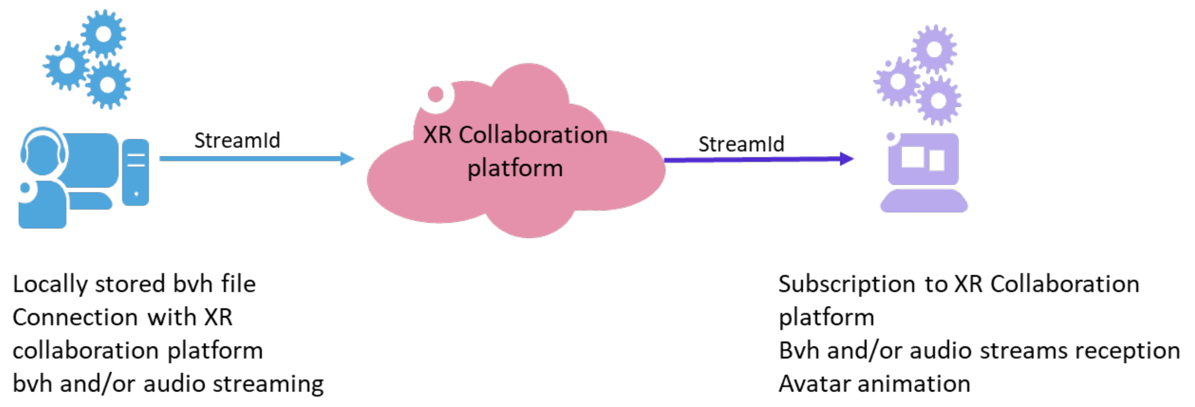

StreamId
XR Collaboration platform
StreamId
Locally stored bvh file
Connection with XR collaboration platform
bvh and/or audio streaming
Subscription to XR Collaboration platform
Bvh and/or audio streams reception
Avatar animation

Figure 5 - Test set-up for Renderer connection to XR Platform

- **Iteration 3**: This step tests the full transmission chain with the transmission of a .bvh stream from the emitter and the decoding and rendering of the kinematics at the renderer side. This iteration enables us to make a first assessment of the transmission delay.

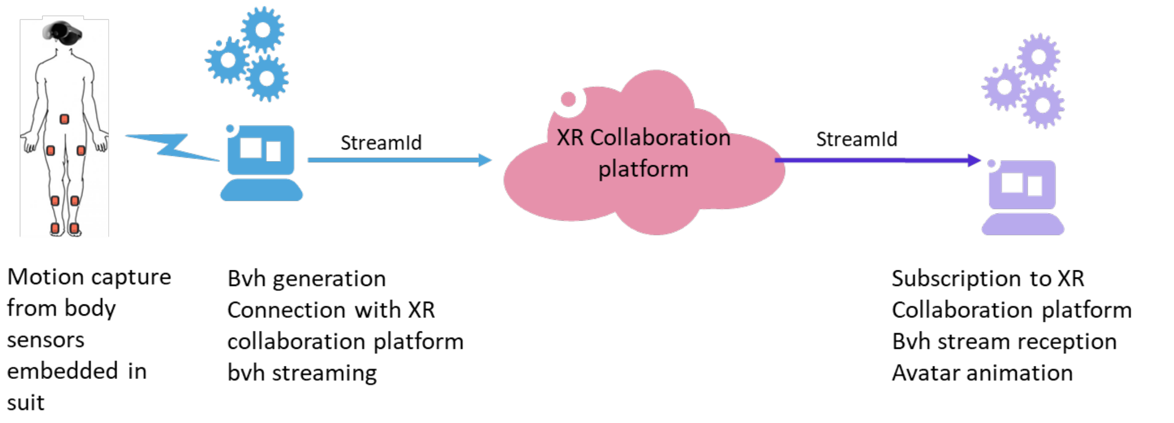

StreamId
XR Collaboration platform
StreamId
Motion capture from body sensors embedded in suit
Bvh generation
Connection with XR collaboration platform
bvh streaming
Subscription to XR Collaboration platform
Bvh stream reception
Avatar animation

Figure 6 - Test set-up for full transmission chain validation

- **Iteration 4**: This step aims at validating the ability of the infrastructure, as designed, to support multi-streams. This iteration is split in two. First, we test the ability of the rendering side to animate two avatars from two different .bvh streams. Then, we validate the ability of the communication infrastructure to route two .bvh streams towards two different renderers. Iteration 4 uses the test set-up described in Figure 1

- **Iteration 5**: This step focuses on the addition of audio to the kinematics stream. It considers again only one emitter with one kinematic stream and one renderer with the animation of a single avatar. It enables us to assess the delay that might occur between kinematics and audio and its impact on the animation of the lips of the avatar. As this animation depends on the voice, audio and .bvh streams must be synchronized to ensure good synchronization of lips movements and body movements. This iteration uses HMD as the audio source.

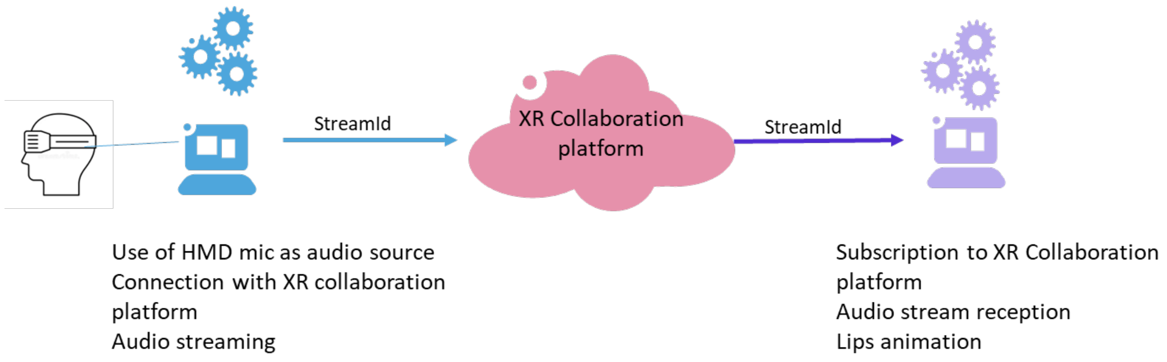
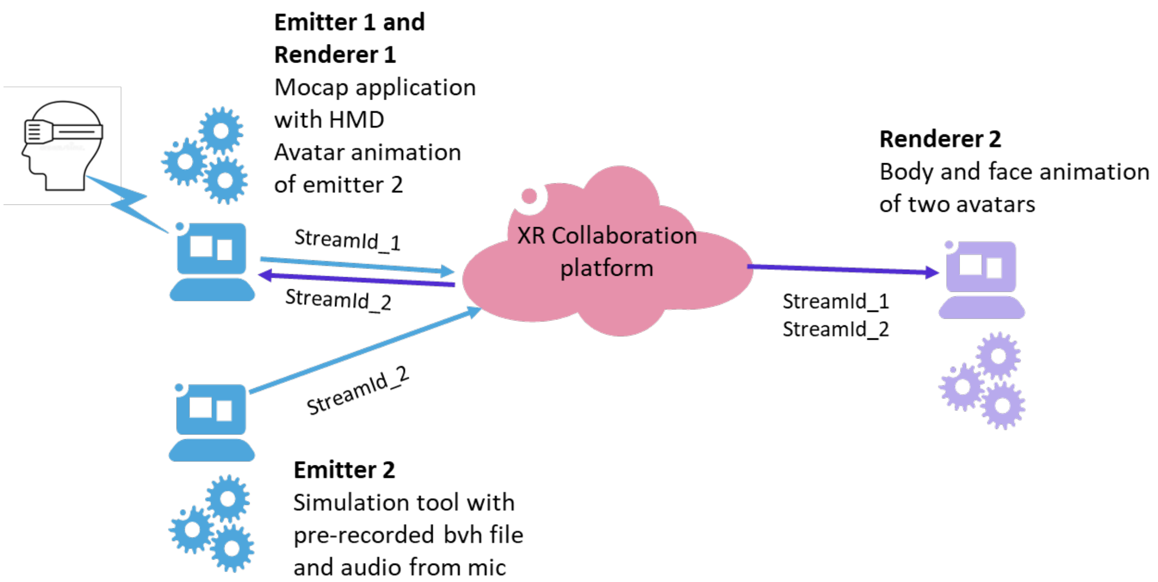

StreamId
XR Collaboration platform
StreamId
Use of HMD mic as audio source
Connection with XR collaboration platform
Audio streaming
Subscription to XR Collaboration platform
Audio stream reception
Lips animation

Figure 7 - Test set-up for audio based avatar lips animation

- **Iteration 6**: This step simulates a real configuration for healthcare pilot where audio and motion capture and rendering will be done with the same headset. It also includes a multi-users configuration. This test assesses the overall performance of the platform from avatar animation accuracy, overall latency, synchronization between streams and CPU consumption. It also validated the capability of the computer controlling the HMD to process both audio and kinematics encoding and decoding and rendering.



Emitter 1 and Renderer 1
Mocap application with HMD
Avatar animation of emitter 2
StreamId_1
StreamId_2
XR Collaboration platform
Renderer 2
Body and face animation of two avatars
StreamId_1
StreamId_2
StreamId_2
Emitter 2
Simulation tool with pre-recorded bvh file and audio from mic

Figure 8 - Test set-up for validation of audio and rendering on same HMD

4 RESULTS

The overall assessment of all tests is qualitative. The pass/fail criteria are the following:

Id	Summary	PASS criteria
1	Kinematics encoding and decoding	Avatar animation nearly synchronous with body kinematics
2	Renderer connection with XR collaboration platform	Avatar animation nearly synchronous with body kinematics
3	Full transmission chain	Avatar animation nearly synchronous with body kinematics and audio quality similar to classical audio conference with platforms like Teams or Rainbow.
4	Multi-users, emission and rendering	Avatar animation nearly synchronous with body kinematics for all avatars and on all rendering devices
5	Lipsync	Audio quality is good enough to have convincing lips animation
6	Audio+Capturing/Rendering on same HMD	Avatar animation nearly synchronous with body kinematics and audio quality similar to classical audio conference with platforms like Teams or Rainbow

Table 3 - Pass/Fail criteria

At this stage of the project there is no criterium regarding the transmission delay neither the synchronization between the various streams.

The test results are summarized in the table below

Id	Summary	Status	Comment
1	Kinematics encoding and decoding	Pass	
2	Renderer connection with XR collaboration platform	Pass	
3	Full transmission chain	Pass	A 500ms delay on audio channel compared to kinematics channel has been measured.
4	Multi-users, emission and rendering	Pass	Usability may be improved to ease the connection setting and automate the arrival of new streamers
5	Lipsync	Pass	Audio quality is good enough to have convincing lips animation
6	Audio+Capturing/Rendering on same HMD	Pass	For the sake of the test, bvh stream is delayed of around 500ms to ensure synchronization between speech and lips animation.

Table 4 - Test Results

Video 1: Full Body Animation

The recording highlights good audio and good rendering from full body mocap and the ability of the platform to route several streams towards several renderers. In the test set-up there are two sources of kinematics, one from a pre-recorded bvh file and one from real-time motion capture. There are also two rendering points where the two avatars associated to the kinematics of the two sources are animated. The video is recorded at one rendering side.
The left part of the video shows the person wearing a suit with motion sensors. The right part of the video shows the animation of corresponding avatar at one rendering side. The avatar mimics the person kinematics.

D5.6_FullBodyAvatarAnimation.mp4
https://www.youtube.com/watch?v=EuujmyVqEeM

Video 2: Lipsync at rendering side

The recording highlights the face animation based on audio captured by the microphone of the HMD. The video showcases the lip animation based on audio transmitted through the XR collaboration platform.

D5.6_LipsyncRenderingSide.mp4
https://www.youtube.com/watch?v=tHaQCQ5rD0E

Video 3: Lipsync and Rendering on same HMD

The recording highlights the face animation based on audio and the ability, in terms of performance, of a single computer to deal with both audio and kinematics capturing and rendering.
The video showcases the rendering viewed in an HMD of the animation of the face of an avatar as well as the audio. Lips animation is based on audio captured on the very same HMD. The audio heard in the video is synchronized to the lips animation as it is the one playback in the HMD.

D5.6_LipsyncRenderingSameHMD.mp4
https://www.youtube.com/watch?v=ERUkxKCftv0

5 Limitations and Next Steps

This first integration step validates the XR collaboration platform architecture for multi-VR user and multi streams functionalities. Several improvements are to be added for the next step. The tests revealed some limitations regarding the audio latency. The solution does not already support face and eye animation based on eye-tracking. And the user experience for setting up the connections should be improved.

- Audio latency: The current integration shows limitation regarding the audio latency compared to the mocap streaming. It is still to be confirmed where this latency occurs while the first experimentation suggests that this is due to an Unreal Engine limitation. This latency not only impairs the real-time experience but also adds issues regarding synchronization between kinematics and audio. New alternatives have to be explored to both add synchronization mechanism and reduce the overall latency of the system.
- Eye-tracking: Eye-tracking is performed thanks to a new opto-electronic system integrated in LightSpace HMD with the support of Ricoh. The eye position is calculated at the source and the information will be transferred for rendering integrated in the already existing bvh streams. The real-time measurements will be used by Golaem for the avatar face animation.
- User Experience: In the current platform status, the users, at the emitting and rendering side, have to agree on common 'Ids' for the streams name and launch manually the connections through the collaboration platform. This poor user experience could be highly improved once the Pixel Streaming infrastructure will be integrated within Rainbow communication platform. This integration will bring a more friendly way for registering users and devices and setting-up the real-time collaboration streams.

Considering our results and the given limitations, our next steps will focus on the three items described above: reduce the audio latency and provide good synchronization between all types of streams, add eye-tracking to enhance the avatar face animation, and improve the overall user experience when setting up a Sharespace real-time collaboration space.