WHOLODANCE

Whole-Body Interaction Learning for Dance Education

Call identifier: H2020-ICT-2015 - **Grant agreement no**: 688865

Topic: ICT-20-2015 - Technologies for better human learning and teaching

Deliverable 6.1

Report on hardware devices adaptation

Due date of delivery: April 30st, 2018

Actual submission date: May 7th, 2018

Start of the project: 1st January 2016
Ending Date: 31st December 2018

Partner responsible for this deliverable: MOTEK

Version: 1.0

Dissemination Level: Public

Document Classification

Title	Report on hardware devices adaptation
Deliverable	D6.1
Reporting Period	M1-M28
Authors	Oshri Even Zohar, Jasper Brekelmans
Work Package	WP6
Security	Public
Nature	Report & demonstrator
Keyword(s)	Volumetric projection, Augmented reality, Mixed reality, Virtual reality, Holography, Dance

Document History

Name	Remark	Version	Date
Oshri Even Zohar		1.0	

List of Contributors

Name	Affiliation
Oshri Even Zohar	MOTEK
Jasper Brekelmans	MOTEK
Georgios Tsampounaris	ATHENA

List of reviewers

Name	Affiliation
Sarah Whatley	COVUNI
Edwin Morley-Fletcher	Lynkeus

Executive Summary

This report includes an integration assessment of the currently available, and upcoming 3D holographic and non-holographic volume display methods and their "fitting factor" to the WhoLoDance context of "staying inside" a virtual body. The report will also include conclusions and a recommendation of which of the evaluated display methods are the best candidate to adopt.

Section 1 introduces the report and lists its objectives. **Section 2** refers to rationales deployed in the decision and prioritizing processes. **Section 3** provides an overview of the specifications of currently available devices related to the dance-learning scenarios and the paradigm of "Staying inside a virtual body": addressing both technical challenges in responsiveness, lag, latency & data-streaming, and creative challenges of usage of such paradigm in a dance teaching and choreography context.

Contents

Executive Summary 3

1. Introduction 4

2. Rationales deployed 4

3. Specifications of available (and upcoming) devices 7

HOLOLENS by Microsoft research 7

LEAP-ONE by Magic Leap company 8

General 8

C API 9

Magic Leap Remote 10

Known Issues 10

General 10

Magic Leap Remote 10

Developer Tools 12

Integration road map 13

Calling Unity web player content functions from the web page: 16

Calling web page functions from Unity web player content 16

Dance teaching and Choreography relevance 18

Bibliography 20

1. Introduction

This deliverable presents an integration assessment of the currently available, and upcoming 3D holographic and non-holographic volume display methods and their "fitting factor" to the WhoLoDance context of "staying inside" a virtual body. The logic of and rationale for the assessment of devices is explained and conclusions / recommendations of the best fit device(s) to WhoLoDance are included. In addition, a glimpse into upcoming research in the areas of new and suitable relevant devices is included. This deliverable is also paired with a demonstrator that has been tested in several partners' labs.

2. Rationales deployed

Which devices are suited and why: The exploration and research into volumetric and holographic projections for Wholodance demanded a framework of rules. Those rules have been created with the intent to make the selected devices adhere with the main aim of the project, namely to be able to have a dance student interact meaningfully with an avatar that is driven by the dance master motions.

Decision factors: VR, AR and MR technologies are evolving at a fast rate, a primary decision factor when aiming at technology that will sustain future hardware developments. We have looked at several

families of devices to find the best fit for the project. We explored technologies coming from VR (Virtual reality), AR (Augmented reality) and MR (Mixed reality). The ground rules were:

- The selected device should be able to stream and display real-time 3D motion capture data driven avatars.
- It should offer a plausible field of view and sufficient resolution to create believable user experiences.
- It should be untethered so the user is free to move in the space.
- It should be able to read and display data streams coming from an external source (either a local graphic workstation or a web server).
- It should be operated from a standard laptop (i.e. not presenting a cost prohibitive issue). *(Note: We have aimed at the assumption of "what will be standard laptops graphic capabilities at the end of 2018)*
- It should not have any noticeable latency or lag in performance. This is critical for real-time user experience.
- Multiple devices (One for teacher / choreographer and a second one for the dance student) are capable to communicate wirelessly with each other in the same 3D coordinate system.
- It should not isolate the user from the real world.

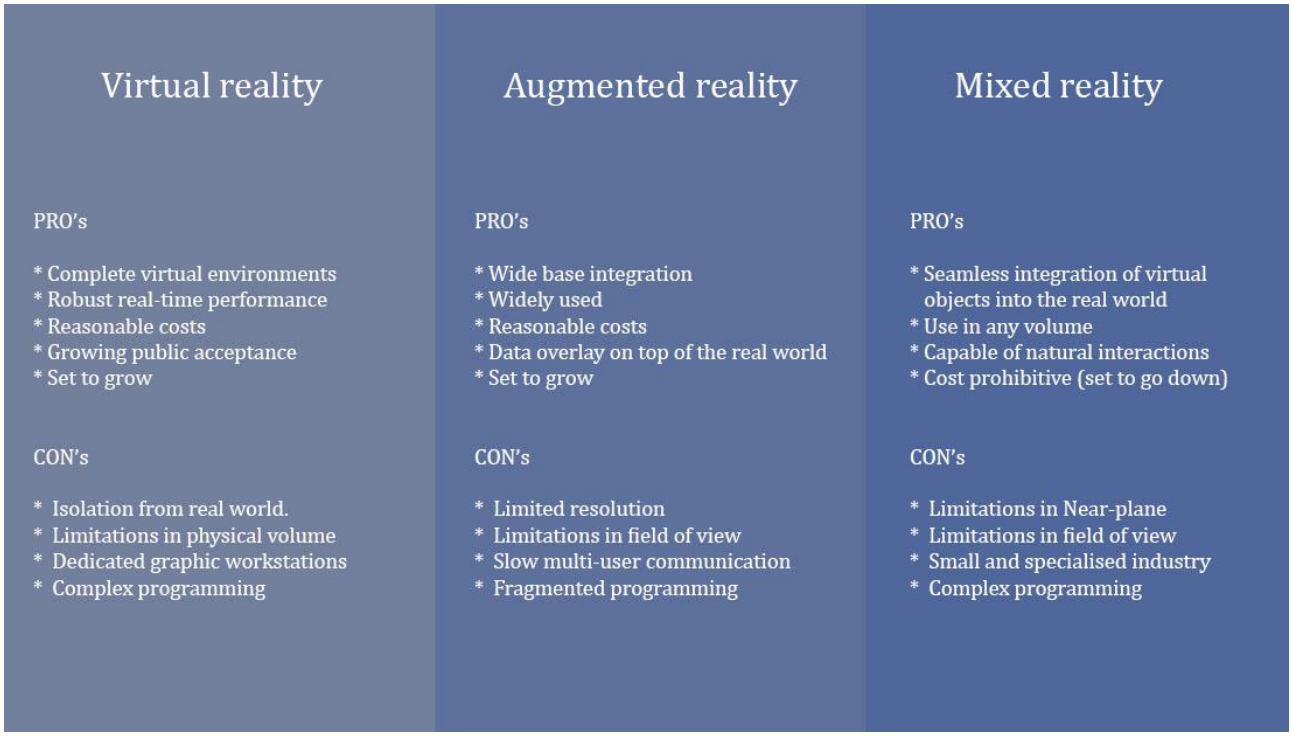


Fig1: Pro's and Con's in VR/AR/MR devices.

The first main conclusion was that in the context of the Wholodance project, VR had to be dropped since the user's isolation from the real world is a fundamental problem for a dancer. We do however keep our attention on upcoming developments in VR that will allow a "Mixed mode" enabling some real world recognition inside a virtual environment.

The second "prime directive" of Wholodance was about achieving proper and robust real-time, full body interaction between the dancer and the avatar in the same volumetric space. That narrowed down the possibilities and led us to focus the search for suitable devices in the MR domain only.

With this in mind, we narrowed down the search for the most suitable devices to reach the goal. We have decided to concentrate on the device that offered the best available specifications while maintaining attention on market developments foreseen to bring to market new devices in 2018. The best candidate was the Microsoft Hololens device.

Fig2: Assembled and disassembled views of the Hololens MR device.

There were two compromises with regard to the Hololens, one had to do with its relatively small field of view compared to other devices and the second one had to do with its near plane (the plane where the virtual image is too close to the eye to be viewed properly). However, other devices had more problematic issues related to their specifications (limitations in frame rate, tracking quality, display resolutions, field of view, lag and latency).

We will continue to review and we are keeping close attention on one more device that is estimated to be brought to market during 2018, and we are developing the Wholodance platform in such a way that it will allow easy migration to the new device. That other device is called "Leap one" from the company Magic leap.

Fig3: Illustrated view of the upcoming Leap one device.

There are no official specifications released of the device yet, but it promises to be better than the Hololens in the two main problems areas mentioned before: the limited field of view and near plane issue.

3. Specifications of available (and upcoming) devices

The comparative specifications of the two devices are listed below. The details about the second device have not been fully released, however an SDK has been released, and some of the mentioned specifications are derived from it.

HOLOLENS by Microsoft research

Display	See-through holographic lenses (waveguides). 2x HD 16:9 light engines. Automatic pupillary distance calibration. 2.3M total light points holographic resolution, 2.5k light points per radian
Sensors	Inertial Measurement Unit, 4x environment understanding cameras, mixed reality capture, 4x microphones, ambient light sensor
Processor	Custom Microsoft Holographic Processing Unit HPU 1.0, Intel 32-bit architecture
Storage	64GB

Ram	2GB
Weight	579g (1.2lbs)
Camera	2MP photos, HD video
Audio	External speakers, 3.5mm audio jack
Connectivity	Wi-Fi 802.11ac, Bluetooth 4.1 LE, Micro-USB 2.0
Power	2-3 hour active use battery life, 2 weeks standby, passive cooling
OS	Windows 10 with Windows Store. Human Understanding: spatial sound, gaze tracking, gesture input, voice support

LEAP-ONE by Magic Leap company

Magic Leap's hardware and software is in a technical preview state right now. Some features or functionalities of the Magic Leap hardware or software may not be available, may contain bugs or other defects, and/or may experience delays or failures. The following section outlines the specifications as published by Magic Leap and we are currently analysing whether or not this is likely to provide an advance on the Hololens. Magic Leap provides guidance on the current state of the hardware and software to help developers make the best possible use of the platform during the pre-release period. Forums on their developer portal also offer more guidance. The following is drawn directly from their published specifications.

General

- SDK directory restructure. If you are using your own build system based on previous versions of the SDK, you may be affected.
    - There are no longer debug and release builds of the libraries as they are just stubs used for linking (e.g. lib/win64/release is now just lib/win64).
    - The directory containing the libraries for device builds is now lib/lumin.
    - The directory containing the sysroot and STL implementations is now the lumin directory at the root of the SDK.
- The mabu target for the device has been changed to lumin. If you are using mabu -t ml1 , change to mabu -t lumin or to mabu -t device. The device target is an alias and is future proof. Any references you might have in mabu project files must be changed as well.

C API

Audio: Stereo audio output and voice microphone recording are supported.

Camera: Capture still images and videos from the color camera.

Dispatch: Allows apps to open URLs using the Magic Leap Browser or other apps.

Eye Tracking: Ability to retrieve fixation point position and eye centers. Blinks can also be detected.

Graphics: OpenGL ES, Desktop, and Vulkan rendering paths.

Hand Gestures & Key Point Tracking: Recognize the user's hand poses (gestures) and track the position of identifiable points on hands such as the tip of the index fingers.

Head Tracking: Headpose is tracked in full six degrees of freedom (DOF).

Image Tracking: Track the position and orientation of specified image targets in the user's environment.

Input (Control / MLMA Support): Retrieve either 3 DOF (orientation) or full 6 DOF (position and orientation) from the Magic Leap Control. Detect button and touchpad presses and the analog trigger. Trigger values range from 0 to 1. A range of touchpad gestures are also supported, as are haptic vibration and LED ring feedback. This interface works seamlessly with both physical Magic Leap Controls and the Magic Leap Mobile App.

Light Tracking: Provides information (luminance, global color temperature) about the ambient light of the user's environment.

Media Codec: Low-level, hardware-accelerated media encoding and decoding.

Media Player: Simple, straightforward media playback interface.

Meshing: Converts the world's depth data into a connected triangle mesh that can be used for occlusion and physics.

Music Service: Supports connecting and listening to a streaming music service.

Occlusion: An interface for feeding depth data to the Magic Leap platform for hardware occlusion.

Planes: Recognize planar surfaces in the user's environment for placing content. This includes semantic tagging for ceilings, floors, and walls.

Raycast: Fire a ray and get the point of intersection with the world's depth data.

Secure Storage: Save data from your app to the device's encrypted storage.

Magic Leap Remote

Supported ML Remote APIs:

- Head pose
- Eye tracking
- Hand Gestures and Keypoint tracking
- Control Input
- World Raycast
- World Meshing
- World Planes
- Image Tracking (Simulator only)
- Graphics API (OpenGL)

- Properties panel in the GUI let you manually set the state of API data such as head pose, detected gestures, and so on.
- Shows the virtual room in a Mini Map as well as an Eye view that depicts your application content composited over the virtual room.
- Supports gamepad, keyboard, and mouse bindings. Use a gamepad to drive head position, define keys to trigger detected gestures, and more.
- The bundled virtual room generator app procedurally generates random rooms that you can customize even further to test your game logic in the Simulator.
- Experimental support for faster graphics rendering in ML Remote can be enabled by setting the environment variable ML_REMOTE_ENABLE_GPU_SHARING = 1. This feature is experimental, because it has not yet gone through rigorous testing and might lead to a crash or visual artifacts under some circumstances.

Known Issues

General

- Stability and reliability improvements to Image tracking are still under way.
- LED colours are not mapped correctly.

Magic Leap Remote

- ML Remote is very resource intensive and currently uses a lot of CPU.
- The ML Remote currently does not report a version identifier.
- The Zero Iteration (ZI) mode in our integration with Unity® software verifies that ML Remote and the Unity® software are in a valid configuration state before starting ZI. This prevents hangs and

crashes. If you see the messages "Virtual Reality SDK Lumin failed to initialize. Attempting to enable None instead" or "No ML VRDevice loaded!" in the Unity® console, consult the documentation for troubleshooting steps.

- The name change from Virtual Device to Magic Leap Remote is not yet fully reflected in the folder names and file layout. We changed the product name to more accurately reflect that your app is running on the host and that a remote, out-of-process simulator of ML1 device is handling the Lumin API requests.
- The properties UI can sometimes lock up. The issue seems to be related to rapid user updates to the property values. The workarounds are: collapse the UI property sections, avoid the "label drag" interface for modifying propery values, avoid using the gamepad to modify property values.
- The Mini Map or Eye View might lock up. Closing and re-opening the Simulator window can fix it.
- The Mini Map/Eye View might start empty in some launch scenarios. Restarting the ML Remote can fix it.
- ML Remote and Package Manager have the same icon.
- Your application might hang or crash if ML Remote crashes while you are iterating.
- Editing the orientation (rotation) properties through the Properties window is currently disabled. Select and rotate objects in the Mini Map view with the rotation gizmo.
- You may only bind alphanumeric keys from a US keyboard layout in the Peripheral Bindings Editor.
- In a multi-display environment, moving the Simulator window to another display can corrupt the eye view if you then resize the window. Resize the window on the original display before moving it to a second display.
- If you enabled the experimental graphics acceleration with ML_REMOTE_ENABLE_GPU_SHARING = 1 in your environment variables, the left eye is rendered for both eyes in the two eye view.
- The Unreal Simulator Meshing sample contains an issue where the length of the confidence array does not match the length of the vertices array.
- The Unreal Simulator meshing contains an issue where the meshing works even if the mesh poll timer is set to 0.
- The gestures confidence value may not be able to be modified at times.
- In Magic Leap Remote, the Input Controller "Controller DOF" setting is changed when iterating on the simulator.
- Renaming the virtual room and restarting causes the toolbar menu option to disappear in the mini map.
- In the ML Remote ☰ menu, clicking "Start Simulator" after clicking "StartMagic Leap Remote Server" does not work. Click the "Start Simulator" button in the ML Remote main window instead.
- When using the simulator, the UI menu stays, even you switch window focus to a different window.

Developer Tools

- VSYNC data in the System Profiler tool cannot be seen on reports when connected to secure devices.
- On macOS, the device label appears as "phaedra".
- It is possible to trigger a timed login lockout if you log on to Package Manager multiple times in succession too quickly.
- ML Remote and Package Manager have the same icon.
- A timeout error can occur if running a large app with Visual Studio and mabu.
- In the Visual Studio plugin, the SDK Path under Debug configuration is not honored unless the path is already added to "Tools > Option".
- mldb logcat has been deprecated. Use mldb log instead to see your application's log activity. Some platform logging is also visible with that command. If you need to report an issue to Magic Leap, use the mldb bugreport filename.zip command. It generates an encrypted zip that you can send to Magic Leap for decryption and analysis, and it contains all the information previously seen in the logcat buffers.
- mldb controller on | off has changed to mldb controller pairing on | off
- Setting a bad MLDB Path can make the Power and Thermal Profiler become unresponsive.
- The Power & Thermal Profiler can hang if you set an incorrect path to your SDK.

Technical comparison factors:

Based on the conclusions from our study of multiple devices other than the two chosen, the important factors that are evaluated (in order of importance) are:

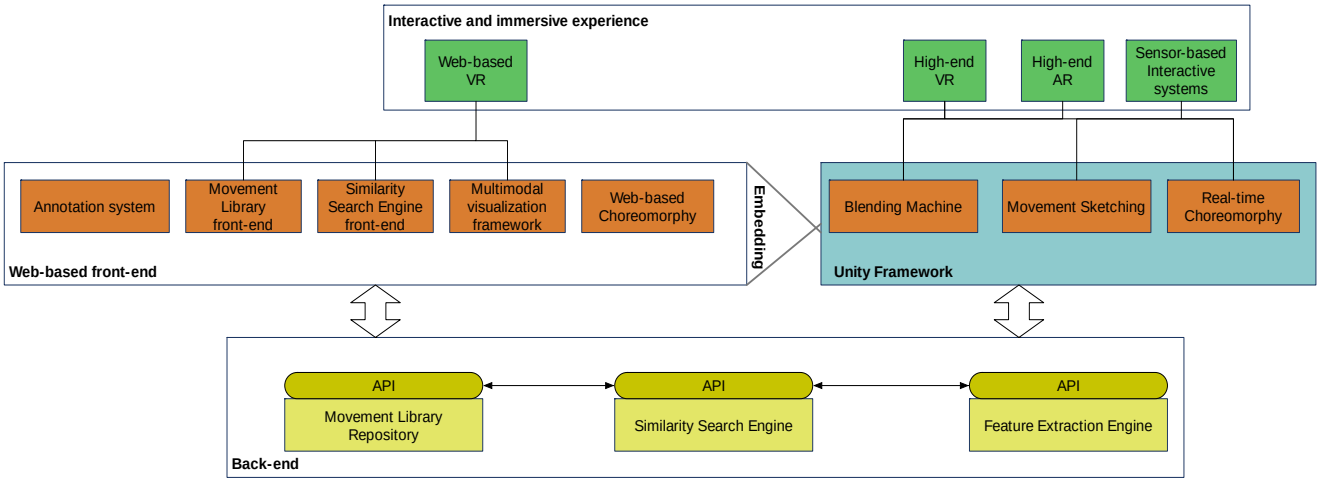
- Robust real-time performance (>30Hz)
- Ability to create a shared anchor and a unified coordinate system for multiple devices
- Tetherless (Communication via WiFi)
- Display resolution
- Field of view

Currently, the available specifications of the 2 devices presented above, suggests that the Leap-one will outperform the Hololens in most areas. However the device is not released yet, and the SDK has been released to selected developers only very recently. After careful analysis of the devices and what is likely to be available within the timeframe of the project, we have decided to retain the Hololens for the duration of the project. Our work in the project is revealing valuable insights to the benefit and limitations of incorporating a head-set within the dancer's environment, which is feeding into knowledge about how dancers learn as well as how head-sets can be developed to take account of user experience.

Integration road map

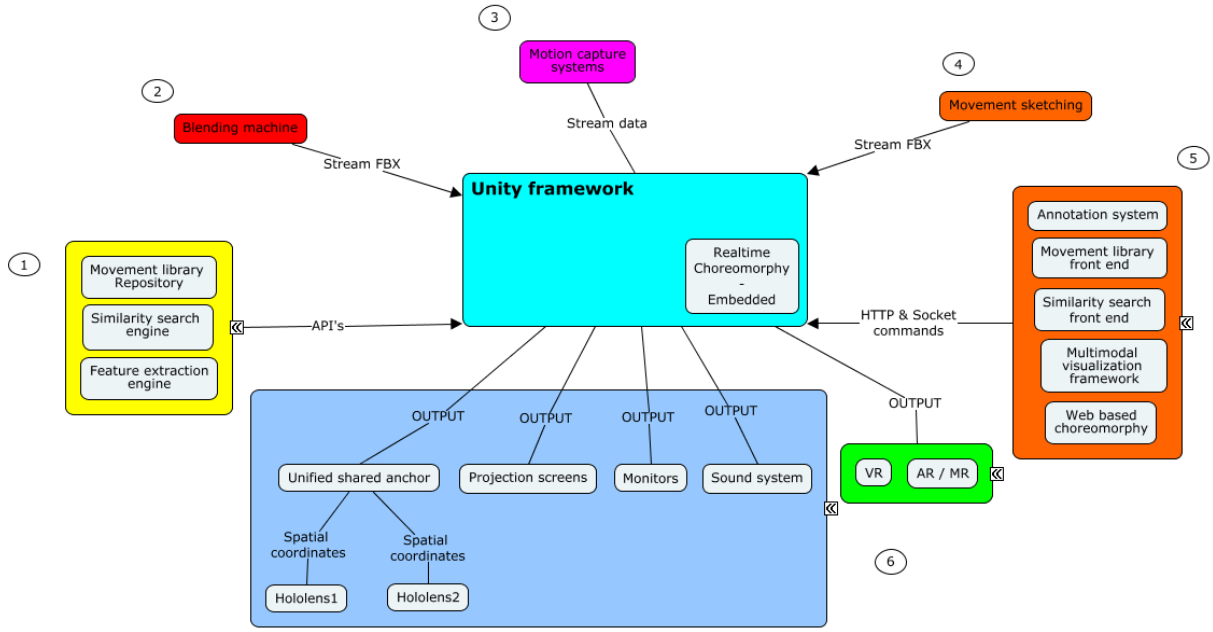
The original integration road map created prior to the choice of the selected device was open to accommodate several devices. After the choice has been made the map became clearer.

Originally it was thought that all partners' individual development components would become native elements inside a single UNITY engine scene. That was initially thought to be the optimal manner to ensure synchronicity between all elements and output a unified real-time stream into the chosen holographic rendering device. However, for different reasons, involving the challenges of developing a range of complex technologies, time planning and skillsets, it was decided to develop a methodology whereby the project tools would be developed as a Wholodance compendium of desktop apps and web based applications. The integration road map was adapted accordingly.

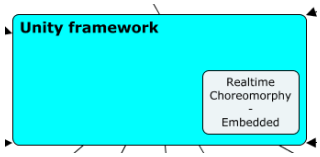


The figure above was presented in D4.3 and describes in general terms the division in application types between all project components.

The figures below describe in detail the types of integration possible for every abstraction layer and their limitations.



The main central block represents the Unity engine and inside it, the only element that is natively embedded into it: Choreomorphy.
The other blocks represents the

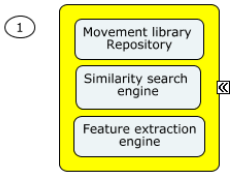


additional project components and
the manner of their integration:

1: **Backend elements**

The process of automating and integrating Backend applications into a custom workflow can be done using Unity's REST API. The API can insight into build history and allows creation of webhook callbacks to facilitate communication between Unity Cloud Build's servers and external applications. The Unity Cloud Build API is based upon Swagger.

The elements that are foreseen to be using this are:



1. **Movement library repository**
2. **Similarity search engine**
3. **Feature extraction engine**

Requests against the REST API are limited to a rate of 100 per minute. To preserve the quality of service throughout Cloud Builds, additional rate limits may apply to some actions. For example, polling aggressively instead of using webhooks or making API calls with a high concurrency may result in rate limiting. It is not intended for these rate limits to interfere with any legitimate use of the API.

The returned HTTP headers for any API request to see your current rate limit status can be checked.

- X-RateLimit-Limit: maximum number of requests per minute
- X-RateLimit-Remaining: remaining number of requests in the current window
- X-RateLimit-Reset: time at which the current window will reset (UTC epoch seconds)

2: **The blending engine**

The blending engine is used to create new FBX files from different sources present in the repository. The users save the new FBX files either locally on a client machine or on the repository server. Those files can be streamed directly into the UNITY engine via TCP-IP or UDP protocols.

3: **Motion capture systems**

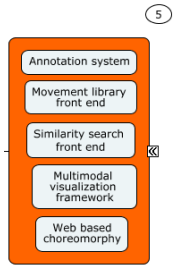
UNITY can read real-time data from a wide variety of motion capture systems. Currently we have tested it with two different kinds of inertial systems. (Shadow motion and Synertial and with the high-end optical system from Vicon.

4: **Movement sketching tool**

The movement sketching tool reads data from low end motion capture devices like depth cameras (Kinect, Xtion, Orbec) or low end inertial sensors (Notch) and provides motion references to other tools like the similarity search tool. This tool can communicate with UNITY as an external stream source.

5: **Web based front end**

Unity supports several methods of web based communication. An HTML page that contains the Unity Web Player content can communicate with that content and vice versa. Basically there are two communication directions:



- The web page calls functions inside the Unity web player content.
- The Unity web player content calls functions in the web page.

Each of these communication directions is described in more detail below.

Calling Unity web player content functions from the web page:

The Unity Web Player object has a function, **SendMessage()**, that can be called from a web page in order to call functions within Unity web player content. This function is very similar to the GameObject.SendMessage function in the Unity scripting API. When called from a web page you pass an object name, a function name and a single argument, and **SendMessage()** will call the given function in the given game object.

In order to call the Unity Web Player's **SendMessage()** function, first get a reference to the Unity web player object. Use the **GetUnity()** function in the default html generated by Unity to obtain a reference to the object. A JavaScript function that would execute the **SendMessage()** function on the Unity web player; in turn **SendMessage()** will then call the function **MyFunction()** on the game object named *MyObject*, passing a piece of string data as an argument.

Inside of the Unity web player content , a script attached to the **GameObject** named **MyObject** is needed, and that script needs to implement a function named **MyFunction.**

A single string, integer or float argument must be passed when using **SendMessage()**, the parameter is required on the calling side. If there is no need for it then just pass a zero or other default value and ignore it on the Unity side. Additionally, the game object specified by the name can be given in the form of a path name.
For example, **/MyObject/SomeChild** where **SomeChild** must be a child of **MyObject** and **MyObject** must be at the root level due to the '/' in front of its name.

Note: GetUnity() might return null if the environment isn't fully loaded yet, so it's a good idea to check if it's value is not null before using SendMessage(). Or wait for the environment to be fully loaded before trying to communicate with it.

Calling web page functions from Unity web player content

In order to call a web page function from within the Unity web player content, use the **Application.ExternalCall()** function. This function calls any JavaScript function defined in the web page, passing any number of parameters to it. It uses the **Application.ExternalCall()**function to call a function named **SayHello()** found within the web page, passing a piece of string data as an argument:

There is in some cases no need to define functions in the embedding web page, instead , use the **Application.ExternalEval()** function to execute arbitrary browser code from the web player content.

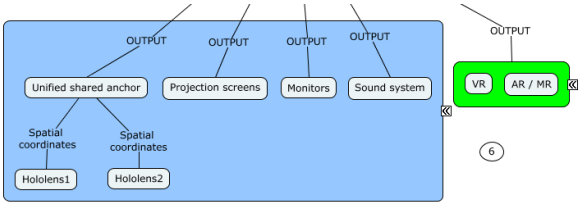
The project components that aim at using web based communication with Unity are:

5.1: Annotation system
5.2: Movement library front end
5.3: Similarity search front end
5.4: Multimodal visualization framework
5.5: Web based Choreomorphy (*)

6: **Outputs**

Unity can output real-time rendered data to a variety of output devices:

6.1: VR
6.2: AR / MR
6.3: Sound systems
6.4: Monitors
6.5: Projection screens
6.6: Unified shared anchor → Hololenses



The main output of the Unity engine in the context of Wholodance will be the holographic rendering output. We aim at streaming output data in real-time to two hololens devices. One will be worn by the dancer / dance student where he or she will be able to see themselves immersed inside an avatar that is driven by motion capture data of the dance masters, coming from the repository. The other hololens device will be worn by the teacher / choreographer and he / she will be able to see the student (live) and the avatar (virtual) overlaid onto one another, and spot the differences in the execution of the dance movements.

In addition, Unity could stream the data to other outputs such as projection screens, computer monitors and additional output devices.

Dance teaching and Choreography relevance

The paradigm of "staying inside a virtual body" in a dance teaching and choreography context presents several usage challenges.

- "One movement fits all?"
    - An interesting question is being introduced here. When we captured the motion data repository, we have used dancers that are considered to be excellent in their respective genres. Their data is used as "motion templates" for the dynamic volumes that the dance students have to "stay inside". The question here is does this volume fit all dance students, and can it be modified to fit all?

- Feedback types (pro or con)?
    - Different avatar models are designed for different interaction types, some present motion volume, and other present suggested (optimal) orientation of body segments. In the rest of the project we also aim to look at advantages and disadvantages of those variables.

- Do technology limitations lead to movement limitations?
    - One conclusion that is present already is that the limitation of the chosen devices will affect the dancer's performance and quality. Having to use the paradigm, while having a limited field of view is one limitation. Latency and lag is another and also the Near-plane intersection of the avatar volume can become a problem.

- Aim to 'stay inside' or 'with' can overtake the focus on self and interior processes of breath, etc.

- The head-set as an apparatus can influence body posture and alignment, which in turn affects the perception of the user's movement in relation to the 'weightless' avatar.

- The one-to-one relationship that isolates the dancer from a wider group

Summary

To sum up the work done in hardware devices adaptation, we have looked at the state of the art of available devices on the market today and filtered out the best suited candidates for usage in the Wholodance project. This filtering work took in consideration the following considerations:

- Future developments and device longevity
- Critical "must have" performance specifications
- Suitability for dance teaching context and choreography
- Ease of use and UI friendliness

We ended up with the preferred present candidate – the Hololens, from Microsoft research. And with a second upcoming device – Leap one, by Magic leap Inc. We have examined in depth the suitability of the chosen device (with regard to the second device, not all information has been released as it is still under development).

We took in consideration also the works done so far by all WhoLoDnce partners and established an integration path for all those elements that would best fit the project's duration, with the intent to create a framework that would easily enable further integration beyond the time scope of the project.

Bibliography

https://www.windowscentral.com/hololens-hardware-specs

https://en.wikipedia.org/wiki/Microsoft_HoloLens

https://docs.microsoft.com/en-us/windows/mixed-reality/hololens-hardware-details

https://www.cnet.com/products/magic-leap-one/specs/

http://www.businessinsider.com/magic-leap-reveals-smart-glasses-photos-2017-12

https://www.engadget.com/2017/12/20/magic-leap-one-details-questions-dont-know/

https://docs.unity3d.com/ScriptReference/

https://docs.unity3d.com/Manual/OfflineDocumentation.html

https://github.com/Microsoft/MixedRealityToolkit-Unity

https://blogs.msdn.microsoft.com/mvpawardprogram/2017/10/31/immersive-apps-mr-toolkit/

https://docs.unity3d.com/Manual/windowsholographic-anchorsharing.html

https://docs.microsoft.com/en-us/windows/mixed-reality/shared-experiences-in-mixed-reality

https://forum.unity.com/threads/unified-coordinate-system-for-string-and-qcar.108256/