A 2D video player for Virtual Reality and Mixed Reality

FILIP MORI
En 2D-videospelare för Virtual Reality och Mixed Reality

SAMMANFATTNING

Medan 360 graders video under senare tid varit föremål för studier, så verkar inte traditionella rektangulära 2D videos i virtuella miljöer ha fått samma uppmärksamhet. Mer specifikt, 2D videoupptäckning i Virtual Reality (VR) och Mixed Reality (MR) verkar sakna utforskning i egenskaper som upplösning, ljud och interaktion, som slutligen bidrar till "presence" i videon och den virtuella miljön. Det här pappret reflekterar över definitionerna VR och MR, samtidigt som den utökar de kända koncepten "immersion" och "presence" för 2D video i virtuella miljöer. Relevanta attribut till "presence" som kan appliceras på 2D video utreddes sedan med hjälp av litteraturen. Det huvudsakliga problemet var att ta reda på komponenterna och processerna i den mjukvaran som skall spela upp video i VR och MR med företagsönskemål och avgränsningar i åtanke, och möjligen, hur man kan justera dessa komponenter för att utöka närvaron i framförallt 2D video och sekundärt den virtuella miljön, även om dessa medium är relaterade och kan påverka varandra. Examensarbetet tog plats på Advrty, ett företag som utvecklar en annonseringsplattform för VR och MR.

Utveckling och framtagande av komponenterna, var gjorda genom inkrementell utveckling där en enklare 2D videospelare skapades, sedan genom en andra inkrementell fas där videospelaren implementerades i VR och MR. Jämförelser med proof-of-concept videospelaren i VR och MR samt den enklare videospelaren gjordes. I diskussionen om arbetet, gjordes reflektioner på användningen av open source-bibliotek i en kommersiell applikation, de tekniska begränsningarna i nuvarande VR och MR Head-mounted displays, relevanta "presence" inducerande attribut samt val av metod för utvecklingen av videospelaren.

A 2D video player for Virtual Reality and Mixed Reality

ABSTRACT

While 3D degree video in recent times have been object of research, 2D flat frame videos in virtual environments (VE) seemingly have not received the same amount of attention. Specifically, 2D video playback in Virtual Reality (VR) and Mixed Reality (MR) appears to lack exploration in both features and qualities of resolution, audio and interaction, which finally are contributors of presence. This paper reflects on the definitions of Virtual Reality and Mixed Reality, while extending known concepts of immersion and presence to 2D videos in VEs. Relevant attributes of presence that can applied to 2D videos were then investigated in the literature. The main problem was to find out the components and processes of the playback software in VR and MR with company request features and delimitations in consideration, and possibly, how to adjust those components to induce a greater presence within primarily the 2D video, and secondary the VE, although the mediums of visual information indeed are related and thus influence each other. The thesis work took place at Advrty, a company developing a brand advertising platform for VR and MR.

The exploration and testing of the components, was done through the increment of a creating a basic standalone 2D video player, then through a second increment by implementing a video player into VR and MR. Comparisons with the proof-of-concept video players in VR and MR as well as the standalone video player were made. The results of the study show a feasible way of making a video player for VR and MR. In the discussion of the work, open source libraries in a commercial software; the technical limitations of the current VR and MR Head-mounted Displays (HMD); relevant presence inducing attributes as well as the choice of method were reflected upon.
A 2D video player for Virtual Reality and Mixed Reality

Filip Mori
KTH Royal Institute of Technology
School of Computer Science and Communication
Stockholm, Sweden
2017

ABSTRACT
While 3D degree video in recent times have been object of research [3] [4], 2D flat frame videos in virtual environments (VE) seemingly have not received the same amount of attention. Specifically, 2D video playback in Virtual Reality (VR) and Mixed Reality (MR) appears to lack exploration in both features and qualities of resolution, audio and interaction, which finally are contributors of presence. This paper reflects on the definitions of Virtual Reality and Mixed Reality, while extending known concepts of immersion and presence to 2D videos in VEs. Relevant attributes of presence that can applied to 2D videos were then investigated in the literature. The main problem was to find out the components and processes of the playback software in VR and MR with company request features and delimitations in consideration, and possibly, how to adjust those components to induce a greater presence within primarily the 2D video, and secondary the VE, although the mediums of visual information indeed are related and thus influence each other. The thesis work took place at Advtr, a company developing a brand advertising platform for VR and MR.

The exploration and testing of the components, was done through the increment of a creating a basic standalone 2D video player, then through a second increment by implementing a video player into VR and MR. Comparisons with the proof-of-concept video players in VR and MR as well as the standalone video player were made. The results of the study show a feasible way of making a video player for VR and MR. In the discussion of the work, open source libraries in a commercial software; the technical limitations of the current VR and MR Head-mounted Displays (HMD); relevant presence inducing attributes as well as the choice of method were reflected upon.

Keywords
Video player, Unity 3D, Virtual Reality, Mixed Reality, Computer Vision.

1. INTRODUCTION
Research into Virtual Reality (VR) and Mixed Reality (MR) in various topics has been going on for quite some time [5] [6], though only recently have these technologies caught the general public’s interest [23]. This thesis looks particularly into playback of video in VR and MR, and focuses on the conventional format of two dimensional (2D) video, that is video rendered in a rectangular flat frame. 360 degree videos (for simplicity, referenced from now on as 360 video), are gaining in popularity, but a non-trivial issue exists with its format. It can be difficult for a movie creator/designer/director to control what the viewer sees. While a viewer can look around anywhere in a 360 video, he/she may miss an important element of the video which the creator/designer/director considers important [3]. 360 video also currently suffers from low resolution [3], which is likely due to the limitations of consumer recording solutions. In all situations where 2D video is needed inside a virtual and immersive environment, a working non-presence breaking video player is trivial for the user experience.

A 2D video player for VR and MR cannot be presented per se, without a virtual environment (VE) for the artifact to exist in. The VE can be viewed as an extension of the real environment, and theoretically only the imagination sets the limits of the contents generated inside of it. Practically, VE is also limited by the capacities of the medium hardware. The resolution of a display is an example of such a capacity and the graphics hardware capacity another. A modern technological medium is the electronic display device such as Head-mounted Displays (HMD) which are the commonly used display devices for VR and MR.

By stimulating the senses of the user using tools of ‘immersion’, a kind of illusion is created so that the user feel present in the VE [8]. These tools here are referenced as technology or communication media. This illusion of being present in the VE resides in the users’ perception and can attention wise be of varying degrees [12]. A degradation of this ‘illusion’ (also known as telepresence) can diminish the experience of watching video content in VR and MR, as the user can lose focus on the video. The video can by itself be regarded as a medium of immersion, although to a lesser degree [8], therefore, sustaining the presence within both the VE and the video can be considered important.

The presentation of 2D videos inside of VEs can be done by applying the motion picture to a texture, which in turn is defined on the surface of an object. The rendering program takes care of the process of the transformation of the surface to the display. A decrease in resolution can be expected when viewing the video from a distance in VE. This does not need to mean a loss of experience or feeling of ‘presence’ compared to consuming video on a two-dimensional screen in the real world, as the user is regardless watching the video from a distance from the screen. The human eye loses precision of distant objects, thus equal sharpness compared to the real world could theoretically be achieved by the hardware and software mimicking the eye limitations.

Games are a big part of the current exploration of VR, which is evident by looking at the software sales [22], thus it is not a coincidence that popular game engines support VR [24]. In this thesis, Unity 3D is used as the front-end environment for the video player and a plugin written in C++ is used for the back-end. Moreover, Unity has support for MR which is beneficial for this work. Unity nowadays has video player back-end functionality of its own and other third party video assets (read plugins/resources) exists in their Asset Store. However, in this paper, comparisons to existing video players are not in focus, rather the purpose is to establish an overview of the components and processes in making a video player for VR and MR, as well as investigating the presence attributions involved. Relatively few papers exist that specifically investigates 2D video playback in VE [1] [9], and none that investigates and documents components and processes concerning 2D video playback for VR and MR using HMDs. This paper attempts to fill in these missing gaps and highlight significant components and processes for 2D video regarding VR and MR.
The VR and MR HMDs used in this work are Oculus Rift CV1 and the development edition of Microsoft HoloLens. Both have characteristic differences and limitations such as pixel density, resolution and field of view (FOV) [13] [20] [21] [23] [25].

2. THEORY

2.1 Mixed Reality

The definition of Mixed Reality can be explored using a concept called the “virtual continuum”. This is represented by a one-dimensional continuous scale with the real environment on the leftmost end of the scale and the virtual environment on the other end. MR is everything between the real environment and the virtual environment. Augmented Reality (AR) and Augmented Virtuality (AV) are also on the scale and included within Mixed Reality. Augmented Virtuality is perhaps a term not many are familiar with, but can be explained by placing a real object (which is then digitalized) inside of VR. Augmented Reality can be explained by placing virtual objects on top of the real world, the real world is therefore “augmented”. [6]

Figure 1 Virtual continuum

The perceived understanding of MR as defined by Milgram & Kishino, is that all kinds of interaction with the virtual world, and vice versa, can be fitted into the definition. [6] The real and the virtual world co-exists, which can be truly evident when viewing the real world through transparent glass with graphical entities being rendered on the screen, while simultaneously interacting with real entities. The theoretical representation of the qualities of the video artifact within MR is supposed to be of no difference in comparison to VR. Rather the difference is of another characteristic, caused by the transparent view of the real world. A greater awareness of the real world allows for more input and interaction of the physical environment.

In terms of AR, rendering of graphics on top of the real world can be a more complex matter than rendering graphics only in VR. Some of the more complicated problems include recognition and scanning of the environment; placement of graphical objects in world space without ‘jitter’ (when graphical objects shake); graphical objects obscuring real objects and vice versa; and achieving pixels that can occlude the real world. For a user to have a good experience and being present within the video in AR, a locked placement of the video frame without jitter in the world space can be considered important as well as correct reflection of colours. For the video, accurate representation of colours can be considered vital for the visual experience.

2.2 Presence, telepresence and immersion

VR is conventionally defined in relationship to hardware. Steuer suggests another way of defining VR by shifting focus from the hardware to the perceptions of an individual [5]. By defining VR in the context of telepresence, which can be interpreted as the experience of presence in a mediated environment (not being the real physical world), more focus can be put on the variables that affect the presence within the environment. His definition of VR is “A virtual reality is defined as a real or simulated environment in which a perceiver experiences telepresence”. The medium creates the possibility of telepresence. The telepresence resides only in the individuals’ head, but the medium still acts a channel to transfer information by the sender and receiver, which could be the same person, or the sender and perceiver being two people or more [5]. Telepresence in this research paper acknowledges the definition Steuer gives, however ‘presence’ will be used instead of telepresence in the remainders of the text, as it should be clear based on context if it is the physical or the mediated environment that is being referred to.

Arguably, 2D video presented within a VE can be VR by Steuers definition. It fulfils the criteria of a real or simulated environment (cartoon or live documentary for example) and it is possible for the user to experience presence within the video. Even if the user experiences presence in a VE, a high-resolution video within the VE could presumably shift most of the attention and presence of the user to the video. The user may be so present in the experience that the rest is acknowledged but blurred out in comparison. Then, the mediated environment which the video resides in, acts as a contributor to the surrounding experience and is important as a medium for the video, but not probably adding to the experience of presence within the video, that should be entirely decided by the characteristics (pixel resolution, spatial and temporal resolution, and contents) of the video.

To be more specific what is meant when the term ‘immersion’ is used, a definition of the word within the framework of this work will be chosen. Slater et al uses immersion as a way of describing a technology, and not to describe the feeling of ‘being there’. Here, the fidelity preservation of senses as opposed to the real world is considered [11]. However, it should be recognized that other definitions exist depending on the context. The definition by Slater et al could be fitted in the context of hardware and software rendering. In the context of psychology, it would not be far-fetched to define immersion as the perception of being in an environment as Witmer et al have [12]. Immersion in games, can be defined in contexts of fast paced action, strategic thinking or storytelling [28]. As a definition that fits the hardware and software rendering is considered most practical for this work, the definition by Slater et al is suitable for this work.

Steuer explores presence by two dimensions he considers relevant, vividness and interactivity. Not to be confused with immersion, vividness can be described as the ability of the technology to provide sensoric information to the user in the qualities of depth and breadth [11]. It is mentioned as one of the attributes providing to the immersion by Slater et al [11]. Vividness is defined by two characteristics, depth and breadth. Depth can be described as the depth of information, e.g. the quality of an image. High resolution for example result in a greater depth. Breadth can be described as the amount of senses provided to experience the environment presented. [5] [11] Commonly, videos are limited in breadth by the audio and visual perception. While it is possible to extend the sensory breadth by adding effects of wind, scent and vibration and other, that is commonly constrained to environments outside of the home or office, such as 4D cinema [5].

2.3 Presence attributions

The following section will describe important attributions to vividness, above all within the depth for 2D video in VR and MR.

2.3.1 Rendering quality and visual acuity

According to Pourashraf, P. et al., significant research has been carried out on the spatial and temporal resolutions of 2D video. What they could not find however, was spatial and temporal analysis’s research performed within video consumption inside of VEs. “To the best of our knowledge, no prior works have investigated the impact of the unique three dimensional characteristics of 3D virtual environments on perceived video quality.” [2] Their study of controlled degradation of video quality, thus saving bandwidth, was evaluated by a user study of Quality of Experience (QoE). By the results, it was evident that down-grading in terms of spatial and temporal resolution was indeed possible when the distance from the video to the user increased in the VE.
In the research on AR and MR, efficiency in regards of bandwidth was of concern in the consumption of 2D video. Among others, vividness is concerned with the resolution and quality of displays [11]. A high resolution adds to the depth of vividness, while a low resolution can diminish the presence. Furthermore, the temporal resolution and pixel resolution of the display technology both adds to the measurement resolution and depth. If the intention is achieving optimal visual video quality, it would be reasonable to look at the limitations of the eye. More specifically, by looking at the eye limitations, it would be possible to conclude whether relevant information of the video, to eyes of the perceivers are lost somewhere between the video, 3d environment rendering and the display technology of today. The following will look deeper into to the pixel and temporal resolutions.

An alternative is to further consider pixel density in VR hardware. The pixel density of consumer HMD’s has yet to match the so-called 20/20 (foot) or 6/6 (meters) eyesight which is regarded as a healthy vision and commonly used as ELR. Moreover, it has been investigated that the 20/20 visual acuity is not accurate enough as display system requirement. The estimation by Lloyd et al. is that the ELR should rather be around 0.5-0.93 arc minutes, for comparison, 20/20 vision is equivalent to one arc minute. One arc minute is defined as 1/60 of a degree and a common pixel density measurement for HMDs is pixels per degree (PPD) which is equivalent to 60/arc minutes. [13] The smallest recommended ELR as defined by Llyod et al. of 0.5 arc minutes is equivalent to 120 PPD. This ELR value, 175 degrees horizontal and 160 vertical monocular visual field [29] would be equal to a resolution of 19200*14400 pixels per eye, a total of 276 Megapixels per eye. This is remarkable comparing to the more common resolution of around 1-2 Megapixels per eye for VR HMDs [23], and so it is possible to conclude that 276 Megapixel resolution VR HMDs are not expected in the nearest future.

By the ELR pixel resolution for HMDs, a user would have no problem perceiving any relevant visual detail of the video artifact presented in the VE. Should the user move backwards however, while looking at the video, eventually minification, as described by Pourashraf, P. et al. [2], would regardless occur, whereas several video texture pixels (also known as ‘texels’) are mapped to a single pixel on the screen, and the video resolution accordingly down-scaled. As the display is capable of a pixel density equivalent to the ELR, the resolution loss should be negligible in comparison to watching video on a flat-screen on distance. However, when minification occurs, the software would need to choose the closest proximal texel or an average weighted color based on proximate texels that is going to be the pixel colour displayed on the screen. Various texture filtering algorithms exists, and what algorithm to choose depends on the computational cost and image quality. A common texture filtering for image processing is bicubic interpolation, which is more resource intensive than a nearest-neighbour algorithm, but renders better image quality [14].

It should be mentioned that there is a difference between perception of information and visual quality. The user may be able to discriminate objects on comparable levels at different spatial and temporal resolutions, but the perceived visual quality by the user can still be higher or lower. As an example of different visual qualities, what can be thought to be a common perplexity is the understanding of temporal resolutions differences of displays, VEs (such as games) and videos. Displays commonly run on a refresh rate higher than 60 Hz to avoid flicker that is a result of display characteristics, and the flicker appearance are different of Cathode Ray Tube (CRT) and LCD (Liquid Crystal Displays). 60 Hz frequencies in CRTs is not sufficient for the so-called flicker fusion threshold for humans, but LCDs uses backlight and flicker does not appear as much on the same frequency. Flicker in displays have a negative effect on the visual quality and can be an annoyance of the user but can be assumed not to normally effect the perception of information as modern LCD screens run at 60Hz or higher. While a temporal resolution of 24 fps for video can be perceived as an acceptable experience, an equivalent frame rate in VEs would not. It is not that the information is greater in the video, but when recording analog or digital video, the photons of light are accumulated over continuous time while the shutter is open, accounting for motion in the image. The final moving images are perceived as smoother because of the motion blur, while the scene in the VE are commonly rendered without accounting for any motion in-between two rendered images. [17]

2.3.2 Attribution of interaction for 2D video in VE

Steuer mentions three contributors to interaction as influencing telepresence, speed; range (amount of interactions); and mapping (how human actions are mapped to actions within the environment). For an effective interaction for video playback within VEs, the focus may lie on satisfactory speed and mapping, while the range of interactions can be limited to play, pause, seek and volume control. The number of interactions can indeed be extended, but these can be considered standard.

The challenges of interaction of 2D within VEs can be explored by studying how interaction and controlling video playback has been done for 360 videos. It can be assumed that the principles of interaction can be adapted to the ones of 2D videos in VE, perhaps in exception from positioning of user interface elements. Pakkanen, Toni et al. concluded that interaction using hand gestures was not mature enough in terms of accuracy and speed, instead the best interaction method for 360 video according to them would be to use a remote control or pointing using head orientation. As mentioned by Steuer, immediate response can make low-resolution games seem highly vivid [5], therefore speed is important, but accuracy, should not be neglected either. With the use of hand gestures, according to Pakkanen, Toni, et al., were problems with the recognition by the system and therefore no immediate response in the environment was made. [4]

2.3.3 Other attributions

Video benefits from higher resolutions, providing to the depth of vividness and immersion. There should be no mistake that digital audio also benefits from higher resolutions (read sample rate and bit depth), contributing to the depth of vividness. The higher the resolution, the better representation of the audio, assuming that the audio was recorded at a high resolution. Moreover, audio is often recorded in more than one (mono) channel. With stereo audio, a different sound can be added to one of the channels, providing an effect that mono is not capable of. This especially is noticeable when wearing headphones. Using stereo audio or more channels, makes it also possible to provide an 3D sound effect.

Three-dimensional (3D) sound is a term used to describe the perception of audio as coming from a certain point in a room. This is valuable for VR and MR because the user would know where the sound comes from. While our visual perception is limited to whatever is in front of us, we can still hear sound from every angle. By locating where a sound comes from we may turn our head and then see (and interact) with the source of the sound. We may also pay more attention to an object that makes a sound even though we can already see it (redundancy). “Sound provides an important channel of feedback that either can be helpfully redundant to a visual cue, or can provide feedback for actions and situations that are out of field of view of the listener” (Begault & Trejo, 2000). [7] Providing the technical basis for sound localisation in a VE would contribute to the presence. An error
in the algorithm, such as the user locating a sound to a source which has no corresponding virtual artifact which should presumably be there, being misplaced, may break the presence. The image and audio aspects of video have been discussed separately previously (See 2.3 and previous paragraph). When the streams finally are demultiplexed and decoded, they need to be synchronized in an efficient manner. While the video container usually has frames per second metadata, the MPEG transport stream as well as other transport stream technologies usually offer a presentation time stamp information for each frame of both video and audio streams. Exclusively relying on frames per second, can make the video and audio go out of sync even though the playback speed of the streams is accurately reflected [15]. One of the effects if the audio and video are not correctly synchronized are lip sync issues. Not seeing the lips move in synchronization with the audio may be, interpreted by the user as not being realistic and presence breaking.

The presence of other individuals in VR is an example of an attribution of presence which does not appear to belong to any specific category [5]. One could imagine, that a rendering of a virtual cinema is made, where several real persons with HMDs sit together, connected and watch a movie. They do not necessarily need to be in the same physical room, but connected through the internet.

2.4 Video decoding and codecs
Raw frames and audio samples are expensive to store on memory and take up bandwidth when being transferred through the internet. Therefore, it is desirable to handle raw video and audio in the shortest amount of time possible, for instance, when the video frame is displayed. Modern digital video playback involves decompressing digital video and audio streams to raw frames and raw audio samples so that software can present them for an audience to view and listen to.

As different codecs exist and new come, it can be convenient if the video playback software supports them without having to make major changes to the software. One way of dealing with this is to use common media frameworks such as FFmpeg, Gstreamer, Apple’s AV Foundation and Microsoft Media Foundation that can be used as an interface between the software and the codec.

This study investigates how a video player can be formed inside of VR and MR, and the variables that effect the presence within the medium technologies. It works its way from the back-end to the front-end of presenting a 2D video player in VR and MR. The question that is asked in this thesis is: What components and processes should be considered when developing a 2D video player for Virtual and Mixed reality? Both software and hardware components have been investigated in previous sections through literature, but the project of practically building the video player can reveal other technical details as well as immersion attributions.

3. METHOD
The main concept is to build a working video player for VR and MR. Furthermore, certain delimitations are taken in to consideration as well as the intention to design the video player following the concepts and attributions of ‘presence’ where applicable, reinforced by immersive technology devices of VR and MR. In the project, two design increments in the development of the video player were performed.

3.1 Development process in software design
With simplicity in mind, to go from the essence of video playback to implementing a video player into VR an MR may require a developing process where the project is split into parts. Incremental and iterative development is a development process, that splits up the work of the intended software into several parts, and manages these parts in incrementing and iterating phases. The definition of incremental development and iterative development appears to be varied and the terms are sometimes also mixed up, however Cockburn makes a clear distinction of the two processes. As incrementing fundamentally means “adding to” and iterating means “do again”, incrementing is used to add features in the software while iterating is used to do rework and adjust the features. They both can be used recurrently in a block of phases, for example the project can start by doing one incremental phase, followed up by a few iterative phases, and end up with a feature completed. Then, the process can be repeated starting with an incremented phase. [16]

The project was divided into two increments. The first increment involved developing a standalone 2D video player. Standalone in this case stands for a single application which plays video (without being included in a 3D environment). This video player was to have basic functionality, which are playing the audio and video (including synchronisation), seeking (go forward or backward) and pausing. When the objective of this increment of finding out and building the video player components was completed, the video player was then re-used and adapted for the second increment.

The second increment involved developing a regular 2D video player for use inside of a VE. Unity 3D, the game engine, was used to incorporate the video player artifact in to the VE. The final stage in this increment involved creating proof-of-concept applications, one for use in VR using Oculus Rift CV1, and one for MR using developer edition of Microsoft HoloLens.

The procedure was to use the standalone video player, modify it and use it as a plugin (back-end) for Unity. Unity would communicate with the plugin to control the video. Then, the interface (front-end), which includes the display of the video and buttons for interaction was constructed inside of Unity. The created video players from the increments where then compared to each other. The objective here was to evaluate the differences between components used.

3.2 Features and delimitations
Advry is a company developing a brand advertising platform for games and apps in Virtual and Mixed Realities. Video playback is a component of the advertising functionality in their upcoming platform, where advertisement can be shown as video.

A video player can be created in many ways with plentiful of features. Therefore, delimitations are important in this work. The company have requests of features as well as delimitations for this video player which are considered.

Company feature requests
- On a modern high-end computer, there should be little to no skipping of frames.
- The video should retain its aspect ratio when displayed inside the 3D environment.
- 3D positional audio. The user should be able to listen to the audio and perceive it as coming from where the video frame is located.
- Attachment of the video frame to an arbitrary surface in the VE.

Of the above feature requests, the thesis had the following delimitations
- 3D positional audio.
- Attachment of the video frame to an arbitrary surface.

Company delimitations
- As there is a time limit for the development of the advertising platform, efficiency and optimizations of the video player are not needed.
• Software libraries may not have a license that disallows linking to proprietary and commercial software.

Other limitations
• Selecting a suitable texture filtering method.
• An effective, but not efficient in terms of resource consumption, video player was considered in first hand. Therefore, no attempts to make it work on a handheld device such as smartphones were performed. A relevant consideration for this project would have been to design the video player for mobile VR such as the popular Google cardboard, mobile MR and/or in contexts where bandwidth is a concern, such as the mobile network when streaming video content.

4. RESULTS
While the back-end can be thought of as a basement machinery, with the purpose of delivering media reflected in the instructions of the media container, the front-end gives a free hand to make the proximate environment immersive and the interface for the video player user-friendly. Most of the focus of the front-end is laid on the interaction necessary to control the video (play, rewind, stop, fast forward, and so on). In Unity 3D, simplicity and using existing techniques were the determining factors that decided the form these interactions took. By pointing using head orientation, and by using load bars or immediate action when hovering over the buttons, clicking with a mouse button or controller was avoided.

In the “Theory” segment, the most significant constraint of the technology was conceivably the pixel density. The assumption is that there is no way of going around this as it is hardware bounded, instead the focus would have to lie on presenting the video container contents accurately as well as relying on other attributes for a positive experience of 2D videos within VR and MR. It was determined that for the video players built in this work, two software libraries were used in the process. A brief introduction to them follows.

FFmpeg is an open-source and cross-platform multimedia library that can be used to decode video and audio. It supports various popular and less popular video formats and codecs.

SDL (Simple Directmedia Layer) is a cross-platform library that provides access to keyboard, mouse, sound card and graphics hardware among other things. It can be used to display video and play audio, and is extensively used in the first increment with minor changes in the second increment.

4.1 Increment one
By following a video player tutorial designed for the FFmpeg library, the first video player was built [27]. The tutorial, although updated as recently as 2015, was not compatible with the latest version of FFmpeg. As it appears, the FFmpeg library is a frequently updated library and several functions were deprecated (as in not recommended to use) in the FFmpeg version used in this work. The tutorial also used an earlier version of SDL (1.2). The code was updated by the tutorial for the first increment. This essentially meant that the latest updates of the libraries were used, which could mean bug-fixes, stability and performance improvements.

A modern full HD resolution display can map each one of the pixels of full HD videos to the pixels of the screen. The video player in increment one uses an example window frame of 640x480 pixel resolution, thus down-scaling was a necessity for HD videos [Figure 2]. Scaling with proper aspect ratio of the video was performed by a SDL library function (SDL_RenderSetLogicalSize) using the dimensions of the video as input parameters. Without applicable scaling of the aspect ratio, the frame would have been stretched out on the vertical dimension, resulting in non-intended presentation of the HD video, potentially mitigating the experience.

Figure 2. A 640x480 window with proper aspect ratio of a full HD video.

The video player is implemented with a video-to-audio synchronisation algorithm based on the tutorial used. At the same time a video frame is displayed, a delay is calculated for when to display the next frame. The calculated delay is a guess based on the time stamp difference between the last frame and the current, and how much the video is behind or ahead the audio. When the delay has been calculated, a timer is set with the calculated value for when to display the new video frame. In the details of the algorithm, measures exist to ensure that the video catches up the audio when being behind it as well as slowing down when far too ahead of the audio.

The core of the delay algorithm is described below and originates from a previous version of the FFmpeg program (which the tutorial is based on) that is a part of FFmpeg [27].

Synchronisation algorithm

\[
\begin{align*}
  a &= (\text{initial delay}) = \text{pts}_{\text{video}} - \text{lastvideo}_{\text{pts}} \\
  b &= (\text{VA difference}) = \text{pts}_{\text{video}} - \text{audiopts} \\
  c &= (\text{estimated delay}) \\
  c &= a \text{ when } (-a) \leq b \leq a \\
  c &= 0 \text{ when } (-a) \geq b \\
  c &= (2 * a) \text{ when } a \leq b
\end{align*}
\]

As shown by the equations, if the audio is approximately close to the current video time stamp, the initial delay (based on the difference of last video frame and the current) is used. Otherwise, the video frames are shown in as fast succession as possible to catch up with the audio or alternatively, delayed times two to allow the audio to catch up.

With the use the arrow keys on the keyboard, it is possible to seek in the video. It is also possible to pause the video by pressing the space key. This can be considered a simple but effective way of interacting with the video.

4.2 Increment two
The main identified problems that were necessary to be solved when implementing a video player inside of Unity 3D were the following, display image data inside of the VE; play audio; and designing interface and interaction. For the interface and interaction, UI elements design and positioning; mapping of
human actions to control the video; and speed of interaction were considered. Finally, the VE was tested on VR and MR devices.

The video player from previous increment was modified to be used as a plugin with a few changes that most importantly allows the video stream to be transferred to Unity. The following describes the main mechanics, of creating a video image in a VE. A vertical plane inside Unity 3D was created. This is the plane where the video texture, generated in code, is eventually mapped to. A pointer to the texture object was sent into the plugin. The plugin has a function where the video frame is attached to the texture and the plugin consequently returns the function (callback) to Unity so that the software can render the texture. By attaching the video frame to a texture and then updating the texture for each frame generated by the plugin, a moving picture is created. Further reference to how to apply graphics in a plugin to a texture in Unity can be obtained from the Unity documentation with link to example code at the bottom of the page [36].

The temporal resolution in videos can be up to 60 frames per second (FPS), or in some cases even higher, but is commonly 24, 25 or 30 FPS. As the video artifact resides in the VE, the rendering frame rate of the VE determines the maximal temporal resolution of the video output, thus the frame rate would need to be higher than for example 24 FPS for the VE to either, not lose frames of the video or slow down the video. With modern hardware and software graphics rendering, the FPS can be stable at 60 FPS in VEs if not more, thus it should not generally be a problem for video residing in VEs, although the video decoding and playback computation itself can contribute to resource consumption and lower frame rate.

4.2.1 Video asynchrony between plugin and VE
As the plugin does not display the video frame by itself, some alteration to the code was made so that the VE could fetch frames from the plugin. The solution did not make any changes to the code of the synchronisation algorithm. The plugin runs the video and audio processes in the background independently of the VE, decoding each video frame in synchronization with the audio. In other words, the VE fetches any currently available frame from the plugin, in an asynchronous manner. For every frame update in the VE, an attempt to fetch a video frame is made. If the VE is faster at rendering a new frame than the plugin producing a frame, then no update to the texture is made. This means that the same video frame can be displayed twice or more, depending on how fast the frame rate is of the VE. A possible negative outcome, is that it can also go the other way, where the frame rate of the VE is lower than the video, resulting in a loss of actual and perhaps perceivable video frames. Depending on how fast the user interprets motion, a user may notice a disturbance in the moving pictures. This solution ensures one important thing, the video decoding process is not dependent on the VE, therefore, sudden heavy rendering in the VE may not disturb the video frame rate. It does not ensure that no frames are skipped, for that several factors such as video resolution, codec and hardware may be dependent upon. However, as the author tested, the playback of a regular H.264 1280x720 resolution video is not a problem perceivably, for a high-end computer with Unity 3D running in VR.

4.2.2 Aspect ratio
In increment two, maintaining the aspect ratio of the video is done through the process of adding black borders (padding) to the texture when the aspect ratio of the plane does not match with the one of the video source. Similar padding functionality to the previous SDL function used in the first incrementation was coded. First, the dimensions of the texture in Unity are set so that they fit the aspect ratio of the plane object. In the plugin, the scaling of the video is done so that the width or height matches respective texture width or height (whichever gets the priority in the plugin to maximise the video resolution). The remaining gaps of the texture width (left and right) or height (top and bottom) is then used as padding (black pixels). It is coded so that the video dimensions (width and height pixels) should fit within the texture, to preserve quality of the video. There are times when the calculated texture size (resolution) becomes too big, in the instance when the aspect ratios of the plane and the video have a great mismatch. This have an impact on the performance, and in those cases the texture resolution is reduced. If the texture size is lower than that of the video, then the video would be down-scaled and resolution and quality would be lost, however the user distance in the VE and the texture filtering explained in Theory are variables that also affect the user’s perception of the video, and so the user may not notice any loss of quality.

4.2.3 Platform-based 2D video player
It can be considered a convenient solution if both the VR and MR could benefit from a uniformly designed interaction. For a VR HMD, the remote control could be a disadvantage, as the user would not be able to see the controller unless rendered as a graphical artifact in the VE. Pointing using head orientation works for both VR and MR HMDs, should they be equipped with the technology of recognizing head movements which both Oculus Rift and HoloLens are. As a result, it can be viewed as a non-presence breaking method.

4.2.3.1 Virtual Reality using Oculus Rift

Figure 3 VR with the video player present.

The video player was implemented in VR using Oculus Rift CV1, and placed in a pre-made futuristic 3D scene. Using keyboard and a controller, the user can walk around in the scene and more importantly, interact with buttons placed next to the video frame. The interaction was made so that the user did not need to press any buttons on the keyboard or the controller. Instead, the tracking capacities of the HMD was used to position the user’s virtual gaze on to the button, whereas an action would be triggered to the back-end. This virtual gaze is not the actual gaze of the user (which would require eye-tracking technology), but is positioned at the centre of the virtual camera. The volume button was made so that a load bar represented the volume in percentage. As the user’s virtual gaze is on the volume button, the bar starts to load up from zero to 100 percent, and the volume is set when the user looks away from the button.
intensify the presence of the VE in none of the devices (which positional audio can do).

Oculus Rift has a seemingly wider FOV than the HoloLens, which makes it possible to view videos up close. However, watching videos at a distance with the HoloLens could still be an experience equivalent to viewing on a real TV or a computer screen (overlooking the PPD), although the user would have to be careful not to move the head around too much which can be a bother. It can also be assumed that even if a large FOV of the video by the virtual camera fits within the VR HMD, the FOV might anyway be too large for the user to enjoy the video.

Some coding was done in the second increment to maintain the aspect ratio of the video on the texture in the VE. This could also be achieved with a library function like the SDL function used in the first increment, and a filter exists in FFmpeg that can accomplish padding. Some extra code was nonetheless necessary on the front-end to maintain the aspect ratio of the texture with the plane object it was attached to. Aside from the visual and audio experience of the video, clearly the interaction mapping of the platform based video player and the standalone player is distinctive. While the fingers were used to press on the keys of regular keyboard to control the video in the standalone player, the VR and MR devices used the tracking of the head, mapped to the centre of the virtual camera to point at the buttons.

### 5. DISCUSSION

Considering that back-end video playback functionality already exists for Unity 3D, the question of why one would want to implement one’s own back-end might be raised. Third-party video player software can save time and money, especially if it is well-made. However, it can come with the drawback of dependency. For example, there is no guarantee that the software is compatible with future releases of Unity 3D, and there is a reliance on the software vendor for support. The native video player is still a new component in Unity and may not have all the required features. By implementing one’s own back-end there is a possibility to re-use the code for other 3D engines as well. Other benefits of implementing one’s own back-end can be achieving better video performance, such as on slower devices which both have to render the VE as well as doing video playback at the same time; and applying filters on videos, before the frame is rendered into the scene.

In the project, a plugin written in the C++ program language is used for the back-end. An advantage of writing code in C++ is that common open-source and cross-platform computer vision and multimedia libraries such as OpenCV, FFmpeg and GStreamer are written in the C-language (and thus compatible with C++). Sometimes bindings of libraries to high-level languages are available, which can be of great convenience and a time saver for a programmer, but it can also be assumed that documentation and support for the native language are better covered, and that the code is more frequently updated. Open-source software is likely to be appreciated because it is free to access and allows users to see and modify code [10]. But open-source software can be an issue in business because of the license (should there exist one) the library comes with. Before incorporating open source code into software for commercial use, looking up potential legal issues should be considered.

### 5.1 Legal considerations using open-source libraries in commercial software

The legal issues of open-source software libraries can be split into two categories: software licences and patents; the latter can be considered more ambiguous and is applied differently depending on the country of jurisdiction. Using open-source software in proprietary (contrary to open source) and
commercialised software can be worth considering despite being troublesome navigating through the legal issues. Avoiding open source libraries altogether may decrease the options of finding a proper library that fits the software. The concern in the following discussion is the legal issues in the commercialisation of proprietary software using open source libraries. To be confident about the legal issues of commercial and proprietary software using open source libraries, the concerned developer or company may ask a lawyer specialised in software for advice.

As common open source software licenses are often bundled with the software and/or hosted on the website of the software provider, access to the license is usually a non-issue. By looking at each library license used in this work, an attempt will be made to determine if the use of the libraries in proprietary and commercial software is viable. Simple DirectMedia Layer 2.0 is covered by the zlib license which allows the developer to use the library for any purpose (a few conditions apply) [30], and will be left out in the discussion. The FFmpeg library used in this work is licensed under the GNU Lesser General Public License (LGPL), which essentially allows commercial and proprietary use of the library as long as the terms and conditions of the license are followed. Some optional parts of FFmpeg are licensed under GNU General Public License (GPL) and cannot be used for closed software to comply with the LGPL license.

One of important condition for the proprietary software to use a LGPL library and keeping the software closed-source, is that the software must fall within the definition "work that uses the library" and not "work based on the library". The end user must also be able to modify and/or relink the LGPL library of the proprietary software. Linking a library to a software can be done in two ways; either the library file is linked statically where the library is embedded into the compiled software (such as the executable), or dynamically linked where the software is linked to a library located outside of the software executable. What could be a common puzzlement is whether statically linking LGPL libraries to proprietary software is allowed. Statically linking is allowed, if the object files of the application are provided [34] so that the user may replace the LGPL library. However it is not always the recommended thing to do [35]. Something that the LGPL has inherited from the GPL is that the distribution of a binary must be accompanied by equivalent source code. That means that if the binary library is bundled with the proprietary software, then the source code of the library must also be made available, such as hosting it on the same server as the proprietary software. If the software uses a library that already resides on the user’s computer, then the distribution of source code is not necessary [34].

A copyright matter is the code used in this project. Although code has been rewritten in the second incrementation, some parts of the code are based on an earlier version of FPlay which is a media player written bundled with FFmpeg. Thus, should the video plugin be used as a proprietary product, then the source code of the plugin (being “work based on the library”) needs to be published as part of the LGPL license. The option if the plugin has to be closed-source, is to remove all copyrighted code and rewrite the missing parts, using other sources (for example, the author’s own ideas or license free code) of inspiration.

To summarise, the LGPL allows FFmpeg to be used in commercial and proprietary (closed-source) software, but the developer/company should be aware of the details of the license such as linking to the library; making sure the software is a “work that uses the library” and when it is necessary to host the source code of the library [34].

Software licenses are often easily accessible and accompanied by open-source libraries at distribution, and various guidance exists to follow the license correctly, such as reading the legal page of the license or license interpretations such as the GNU FAQ [34]. Software patents on the other hand can be an ambiguous issue for a software product publically released because there is likely no way to truly find out if the software infringes on the vast number of granted patents. But there are some more obvious patents within multimedia technology, covering the audio-video codecs. For example, the perhaps most popular video codec of today, H.264, has multiple patents and the right to use this patented codec commercially essentially requires a patent license for distributors of the codec.

Continuing with the applicability of software patents, the law differs from depending on geographic location. Software patents are recognized in some countries such as the US and Australia [31], while in Europe, software patents are not recognized as such [32]. However software patents may be granted in Europe if the invention is of a technical nature [33]. But as law interpretations differs, exceptions exist and the distribution of software on the internet is global, the issue of software patents may still be of a concern. Thus, while there might not be an issue to distribute a decoder of H.264 to a receiver in Europe without proper patent licensing, acquiring a license for the decoder can be considered better than taking the risk of possible patent infringement.

The codec patent issue can be avoided for a developer/publisher of a multimedia software by using libraries that do not themselves contain an implementation of a codec, but instead uses the codecs that are installed on the user’s computer. The drawback of this can be the non-availability of certain codecs. FFmpeg directly incorporates codecs in the library and thereby can be exposed to patent issues should it be used with a commercial software. As an open-source project, FFmpeg is not a patent licensee of the H.264 codec, thus it is up to the developer/company of a proprietary software to acquire a license should H.264 be used and distributed. If H.264 is not necessary for playback, FFmpeg may be compiled without it. Essentially, FFmpeg can be compiled with a very minimal configuration where all non-free codecs are removed. Presumably, this should remove the risk of codec patent infringement in a distributed compiled binary of the FFmpeg library.

To summarize the patent issue, measures to avoid the risk of patent intrusion can be performed on the open-source multimedia library FFmpeg for use in a proprietary and commercial software. This can be to acquire a proper license for codec patents or to compile FFmpeg without the non-free codecs. As a final note to the legal considerations, the focus of these paragraphs has been to discuss the possibility of incorporating the libraries used in the prototype in a commercial product, and not the details how to comply with licenses and patent laws. There may be aspects that have not been covered, and legal counselling may be the only way to confident about the legal issues.

5.2 Open source, economical, ethical and environmental aspects

In this project, open-source libraries were used in the development of the video player for VR and MR. Open source software are free to access and allows user to see and modify code. With this comes several benefits that are highlighted here. To begin with, the freedom to see the source code and modify it can give open-source software an advantage over proprietary code. In open-source projects, people can report bugs, modify and add code or contribute in other ways to the project, whereas in proprietary software, it is not possible to contribute to the project code as an external user. With proprietary software, because the code is closed, it is possible that the same algorithms are reinvented by different organisations and companies. Proprietary software are also more exposed to malware, whereas this is more difficult in open-source software because anyone can view the code. As proprietary software does not reveal its source
code, it is harder to determine the security and privacy of the software. Therefore, open-source software can be considered a more ethical choice than proprietary software. [38]

Open-source software is commonly free to download and use (it is also possible to sell open-source software with licenses such as the GPL), thus allowing consumers that may not be able to buy software to benefit from open-source software.

5.2.1 Efficiency
The plugin in this work is multi-platform and has been tested for both Mac and Windows. From an economical point of view, building software that is multi-platform is beneficial in the way that it saves time and money (both for solo developers and companies), because less coding is needed to support several platforms and maintaining the code becomes easier.

Efficient coded software can reduce the CPU usage and may finally have an impact on the environment as less energy is needed. For example, the battery on the laptop may last longer because more efficient code has been used or been written. Decoding compressed videos is usually the most CPU-intensive operation in digital video playback, therefore improvements in algorithms may reduce the CPU usage and energy consumption, and possibly also benefit the environment. For example, FFmpeg have improvements in the decoder of the free codec VP9. That is more efficient and faster than the native decoder [37].

5.3 Presence attribution considerations and technical limitations
A few aspects of the attributes of video quality, positional audio and interaction, leading to an increased presence in 2D videos inside of VR and MR are considered in this work. Namely,

- the density of pixels of the display as reflected by the eye-limited spatial resolution (ELR)
- the preservation of the dimensions (aspect ratio and optimal Field of View)
- temporal resolution
- interaction alternative
- accurate colour reproduction by the display

Two aspects are delimited in the work

- solutions to positional audio
- texture filtering of the video texture in the VE

Oculus Rift can display a resolution of 1080×1200 pixels per eye while the HoloLens has a maximum resolution of 1280x720 per eye [23][25]. Full HD movies are characterized by the resolution of 1920x1080, and thus it is not possible for the mentioned HMDs to display all the pixels of the full HD movies let alone 4K movies, at any point in the virtual space, even if the video occupies the whole FOV. By adding the resolutions of the stereoscopic displays, Oculus Rift could support full HD movies, but as each lens partly overlap each other’s FOV, giving a stereoscopic illusion, the effective resolution would still be too low [20]. As the pixel resolution is directly linked to the depth of vividness, and the more pixels the greater depth can be achieved, the HMDs used in this project were limited in this property of immersion.

The FOV is a contributor of immersion in VR displays in the extent it surrounds the user [11]. The maximal Oculus Rift CV1 horizontal binocular FOV (overlapping considered) has been estimated as 94°. In comparison the binocular FOV of the eyes is 200° [20] [29], the FOV of the Rift is still enough for it to function in the peripheral vision. With the developer version of HoloLens, the FOV is certainly limited and is roughly 30° horizontally [21]. It is little, but it should be noticed that as the user turns his/her head, the brain can compensate and fill in the gaps where the eyes cannot see. Both HMDs are limited in their FOV (the HoloLens more than the Oculus Rift) considering that they do not reach the binocular FOV of the eyes, which is a limitation in the immersion they provide.

For an acceptable video experience, a limited FOV and resolution can be unsatisfactory. In the case of a limited FOV, the video frame would be cropped if the user were too near, thus the user might need to increase the distance to the frame. How the pixel resolution affects the perception is dependent on several factors, including pixel density, the visual acuity of the individual, brightness of the screen [13] and how the brain makes up for the lack of pixels.

The preservation of dimensions of the video in the VE as perceived by the user is presumably dependent on several factors. Two are considered here. Firstly, the correct aspect ratio as instructed by the video container should be maintained on both the texture and the video frame artifact. Loss of proportions can result in perception of stretched out image frame on one of the dimensions. Secondly, the FOV of the user to the display should ideally be equivalent to the FOV of the virtual camera. Divergence may result in distortion [18] and skewed perception of the image, thereby looking unnatural to the human eye. As the FOV of a conventional display varies depending on the distance from the user, the FOV within the VE would have to be variably changed to avoid any kind of distortion. With the use of a HMD, the FOV is near static as the eyes are always at the same distance from the displays. This makes it possible to use an approximate constant for the FOV for the virtual camera, and thereby the intended representation of the VE as well as the video can be shown.

In terms of VR displays, there is a need of a high refresh rate to avoid the perception of flicker (perception of the screen flashing) and motion sickness. One reason for these perceptually negative effects is that the human eye can detect motion better in the peripheral vision, where the VR usually operates. Therefore, it can be assumed that it is no coincidence that the Oculus Rift has a relatively high refresh rate of 90 Hz. The effective HoloLens refresh rate is 60 Hz; however, it has a limited FOV mentioned earlier and the visuals of the device do not reach the periphery. Therefore, the need for a higher refresh rate is reasonably not as obvious for the HoloLens. [19] [26] As the frame for 2D videos is commonly placed centred in front of the viewer and at a reasonable distance, the peripheral vision can be considered not be so important for the video, as for the VE the video resides in. Significant motion in the peripheral vision in the VE could be a distraction and an annoyance for a user viewing videos.

To keep the temporal video resolution continuous and synchronized to the audio, a video-to-audio synchronisation algorithm was used in the back-end. The synchronisation of video and audio can be considered important to avoid problems such as lip sync errors. To see events happening on screen, but not be able to connect them to the audio being presented, can be presence breaking. Furthermore, the process of timing the video delay (when to display the next frame) was run in the background, independently of the front-end (VE), so that the video frame retrieval by the VE was not synchronised with the back-end. In other words, the back-end offers a decoded frame for a limited amount of time (usually equivalent to the frame rate). Should the VE miss the opportunity to fetch this new frame during that time, then it is lost and not shown. Keeping a higher VE frame rate update than the video is important for this not to happen and the user not losing relevant information (lower temporal resolution) that affect the depth of vividness.
A fitting interaction solution for VR and MR was implemented. By only using HMD orientation tracking, there was no need not grab a controller. In VR, it is an advantage as the HMD occludes the real space and therefore it can be a hassle to grab a controller. In MR, HoloLens supports clicking in the air using the index finger, but pointing using head orientation is better in the accuracy and responsiveness, that are important for the interaction property of presence. For a uniform solution, HMD orientation tracking can be considered a good interaction solution as well, as both Oculus Rift and HoloLens support it.

The audio experience of 2D video in VE can be assumed to be similar to audio listening of a conventional video system in a physical room and it offers more diversity as opposed to the physical limitations of the real world. The point would be to transmit audio in a similar way audio is transferred through air in real life, contributing to the depth of vividness in the video, depth in the VE, or to some extent, both. Two solutions of channelling sound to the listener based on positional audio can be considered, being different from the stereo perception solution presented in this work. They both assume that the user is in possession of stereo speakers, surround speakers, or headphones, and that the media source contains stereo audio.

The first solution would be to take the position of the audio artifact in the VE into consideration and place a mono virtual speaker by the 2D video frame. The audio out of the speakers/headphones simulates a 3D effect where the user can identify the origin of the source, thus rendering a better sense of presence with the VE, but not taking advantage of the stereo perception solution presented in this work, which reflected stereo audio of the video but overlooked the VE.

A second solution is based on the way a stereo TV works. Not all TVs work the same, but many give the user the option of rendering stereo audio in two speakers at each side of the TV. As a solution, two audio sources, each being mono and representing the left and right channels of the native audio source, can be placed on each side of the video frame in VE akin to a conventional TV. The distance from each source is considered and the audio reflected in the ears of the user accordingly. This way, the user can identify the location of the source, and some stereo effect is perceived which can induce a more realistic effect of a TV in a physical room compared to previous methods.

5.4 Method reflections

The incremental build model was used in the making of the video player. It was a suitable choice in the way that it made it possible to test the greater part of the back-end in a native application prior to implementation in the VE. Direct implementation in the VE by use of the waterfall model, for example, would reasonably have made it harder to debug. As an example, both the decoding of audio and video and attachment of moving pictures to a texture in the VE would have to be solved in the same phase. However, a benefit of doing so, would have been that the steps of implementing a front-end for the native application could be disregarded, thus time could potentially have been saved. As the implementation of interaction and interface was relatively simplistic (and the back-end functions were not) in the first increment, little time would have been saved.

Finally, more technical details and algorithms could have arguably been included in the results and later discussed, such as seeking and pausing. However, they are nonetheless related to general video playback, and not necessarily specific for VR and MR.

6. CONCLUSION

The results show a feasible way to implement video playback in VR and MR together with the technical components being significant.

A few variables can be particularly considered when implementing a video player software for VR and MR. These include considering how the interface and interaction should be designed, may depend on the state of the interaction technology and what means of interaction the user has in possession, for example controllers, keyboard, mouse or just the HMD tracking the head. An easy to reach solution of today’s technology would be to assume that the user is not in possession of anything more than the HMD tracking, and thus build the interaction around the tracking ability of the HMD exclusively.

In general, this research project shows that when implementing video playback in both front-end and back-end for the VE, regardless of display device, other variables may need attention, such as the resolution of a texture; aspect ratio; positional audio or stereo audio; synchronization solutions; and interaction and controlling the video, and they may all be important for the presence within the video and/or the VE. A subject of further research is the evaluation of an appropriate texture filtering function.

The company delimitations and feature requests of the video player were considered in this work. Coding solutions for preservation of aspect ratio of the video in the VE and playback of video frames were investigated during the development of the video player. The use of the libraries in a commercial and proprietary software was discussed, regarding patents and licenses that should be considered before deciding to implement the libraries in a commercial software. The remaining feature requests of sound localization (3D audio) and attachment to a surface are subjects of further investigation.

Finally, the developers and designers may need to recognize that the transparent graphics of a MR HMD can disrupt the experience of the video. Measures to counter this might be considered, such as recommending the user to move to a darker area, or increase the brightness/intensify the colours of the screen if possible. This is also a suggestion of future research.

7. REFERENCES


**Figures attributions**

Figure 1 “Reality-Virtuality Continuum” by Giovanni Vincenti is licensed under Creative Commons Attribution 3.0 Unported. Figure 2-6 “Big Buck Bunny” by Blender Foundation is licensed under Creative Commons Attribution 3.0.