An XR rapid prototyping framework for interoperability across the reality spectrum

Efstratios Geronikolakis

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science and Engineering

University of Crete

School of Sciences and Engineering

Computer Science Department

Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Associate Prof. George Papagiannakis

UNIVERSITY OF CRETE

COMPUTER SCIENCE DEPARTMENT

An XR rapid prototyping framework
for interoperability across the reality spectrum

Thesis submitted by

Efstratios Geronikolakis

In partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: Γερονικολάκης Ευστράτιος

Efstratios Geronikolakis

Committee approvals: GEORGIOS PAPAGIANNAKIS Digitally signed by GEORGIOS PAPAGIANNAKIS Date: 2020.09.30 14:00:25 +03'00'

George Papagiannakis

Associate Professor, Thesis Supervisor

CONSTANTINOS STEPHANIDIS Digitally signed by CONSTANTINOS STEPHANIDIS Date: 2020.09.30 10:31:41 +03'00'

Constantine Stephanidis

Professor, Committee Member

DIMITRIOS PLEXOUSAKIS Digitally signed by DIMITRIOS PLEXOUSAKIS Date: 2020.09.30 12:27:15 +03'00'

Dimitris Plexousakis

Professor, Committee Member

Departmental approval: Polyvios Pratikakis Digitally signed by Polyvios Pratikakis Date: 2020.10.02 06:02:40 +03'00'

Polyvios Pratikakis

Assistant Professor, Director of Graduate Studies

Heraklion, September 2020

An XR rapid prototyping framework for interoperability across the reality spectrum

Abstract

Applications of the Extended Reality (XR) spectrum, a superset of Mixed, Augmented and Virtual Reality, are gaining prominence and can be employed in various areas, such as virtual museums. Those incarnating a virtual museum are considered digital heritage applications and are of utmost importance to the preservation of cultural heritage. Unfortunately, the majority of them are used to operate only in one of the above realities. For instance, we notice many applications that exist for Virtual Reality, which cannot be found for Augmented Reality mobile devices.

The lack of virtual museum applications across the XR spectrum is a real shortcoming. There are many advantages resulting from this problem's solution. Firstly, releasing such an application across the XR spectrum could contribute to discovering its most suitable reality. Moreover, it could be more immersive within a particular reality, depending on its context. Furthermore, by releasing such an application across the XR spectrum, its availability increases to a broader range of users. For instance, if it is released both in Virtual and Augmented Reality, it is automatically accessible to users that may lack the possession of a Virtual Reality headset, but not of a mobile device (capable of supporting AR). As a result, the preservation of cultural heritage increases rapidly.

The question that arises at this point, would be "Is it possible for a full s/w application stack to be converted across XR without sacrificing UI/UX in a semi-automatic way?". It may be quite challenging, depending on the architecture and application implementation.

Through our work, we encountered such challenges as well, in two different situations. Specifically, the transition of a virtual cultural heritage playground application from Virtual to Mixed Reality and respectively, of a virtual museum application from Augmented to Virtual Reality. We performed a manual XR transition in these cases, noting the critical steps needed alongside this procedure. As a result, we attempt to overcome this challenge utilizing our XR Transition Framework that we created based on our findings and will present in this thesis.

We present our framework, the "XR Transition Manager", in the context of digital heritage applications (virtual museums). It is an XR transition library, able to manage the underlying software stack for different platforms or realities across the XR spectrum, depending on the developers' choice. Through a simple user interface, developers are able to set their preferences. Specifically, this framework automatically allows transitions across the XR spectrum between Virtual, Augmented and Mixed Reality. It also reduces the development time while increasing the XR availability of digital heritage applications, encouraging developers to release them across the XR spectrum, thus contributing to the preservation of cultural heritage.

Ένα σχεδιαστικό πρότυπο εφαρμογών για διαλειτουργικότητα σε όλο το φάσμα Εκτεταμένης Πραγματικότητας.

Περίληψη

Οι εφαρμογές του φάσματος Εκτεταμένης Πραγματικότητας (ΕΠ), ένα υπερσύνολο Μεικτής, Επαυξημένης και Εικονικής Πραγματικότητας, αποκτούν εξέχουσα θέση και μπορούν να χρησιμοποιηθούν σε διάφορους τομείς, όπως εικονικά μουσεία. Αυτές οι οποίες ενσαρκώνουν ένα εικονικό μουσείο θεωρούνται εφαρμογές ψηφιακής κληρονομιάς και έχουν ύψιστη σημασία για τη διατήρηση της πολιτιστικής κληρονομιάς. Δυστυχώς, η πλειονότητα αυτών συνήθως λειτουργεί μόνο σε μία από τις παραπάνω πραγματικότητες. Για παράδειγμα, παρατηρούμε πολλές εφαρμογές που υπάρχουν για συσκευές Εικονικής Πραγματικότητας, οι οποίες δεν δύναται να λειτουργήσουν σε κινητές συσκευές Επαυξημένης Πραγματικότητας.

Η έλλειψη εφαρμογών εικονικών μουσείων σε όλο το φάσμα ΕΠ αποτελεί πραγματικό μειονέκτημα. Υπάρχουν πολλά πλεονεκτήματα που προκύπτουν από τη λύση αυτού του προβλήματος. Πρώτον, η κυκλοφορία μιας τέτοιας εφαρμογής σε όλο το φάσμα ΕΠ θα μπορούσε να συμβάλλει στην ανακάλυψη της πιο κατάλληλης πραγματικότητάς για αυτήν. Επιπλέον, ανάλογα με το πλαίσιό της, θα μπορούσε να είναι πιο συναρπαστική σε μια συγκεκριμένη πραγματικότητα, συγκριτικά με κάποια άλλη. Επιπροσθέτως, κυκλοφορώντας μια τέτοια εφαρμογή σε όλο το φάσμα ΕΠ, η διαθεσιμότητά της αυξάνεται σε ένα ευρύτερο φάσμα χρηστών. Για παράδειγμα, αν γίνει διαθέσιμη τόσο για Εικονική όσο και για Επαυξημένη Πραγματικότητα, είναι αυτόματα προσβάσιμη σε χρήστες που ενδέχεται να μην έχουν στην κατοχή τους μια κάσκα Εικονικής Πραγματικότητας, αλλά να έχουν μια κινητή συσκευή (ικανή να υποστηρίζει Επαυξημένη Πραγματικότητα). Ως αποτέλεσμα, στο πλαίσιο εφαρμογών ψηφιακής κληρονομιάς, επιτυγχάνουμε την διατήρηση της πολιτιστικής κληρονομιάς.

Το ερώτημα που προκύπτει σε αυτό το σημείο, θα ήταν «Είναι δυνατόν μια ολόκληρη στοίβα εφαρμογής λογισμικού / υλικού να γίνει διαθέσιμη στο φάσμα ΕΠ χωρίς να θυσιάζεται η εμπειρία του χρήστη, με ημιαυτόματο τρόπο;». Μπορεί να είναι αρκετά δύσκολο, ανάλογα με την αρχιτεκτονική και την υλοποίηση των εφαρμογών.

Μέσα από τη δουλειά μας, αντιμετωπίσαμε τέτοιες προκλήσεις, σε δύο διαφορετικές περιπτώσεις. Συγκεκριμένα, τη μετάβαση μιας εφαρμογής εικονικού πειραματισμού πολιτιστικής κληρονομιάς από Εικονική σε Μεικτή Πραγματικότητα και αντίστοιχα, μιας εφαρμογής εικονικού μουσείου από Επαυξημένη σε Εικονική Πραγματικότητα. Συνεπώς, πραγματοποιήσαμε μια χειροκίνητη μετάβαση στο φάσμα της ΕΠ σε αυτές τις περιπτώσεις, σημειώνοντας τα κρίσιμα βήματα που απαιτούνται κατά τη διάρκεια αυτής της διαδικασίας. Ως αποτέλεσμα, προσπαθούμε να ξεπεράσουμε αυτήν την πρόκληση χρησιμοποιώντας τη δομή μετάβασης ΕΠ που δημιουργήσαμε, με βάση τα ευρήματά μας, το οποίο και θα παρουσιάσουμε σε αυτήν την εργασία.

Συγκεκριμένα, παρουσιάζουμε τη δομή μας, το «Διαχειριστή Μετάβασης ΕΠ», στο πλαίσιο εφαρμογών ψηφιακής κληρονομιάς (εικονικά μουσεία). Πρόκειται για μια βιβλιοθήκη μετάβασης ΕΠ, ικανή να διαχειριστεί την υποκείμενη στοίβα λογισμικού για διαφορετικές πλατφόρμες ή πραγματικότητες σε όλο το φάσμα ΕΠ, ανάλογα με την επιλογή των προγραμματιστών. Μέσω ενός απλού περιβάλλοντος χρήστη, οι προγραμματιστές μπορούν να ορίσουν τις προτιμήσεις τους. Συγκεκριμένα, αυτό το πλαίσιο επιτρέπει αυτόματες μεταβάσεις στο φάσμα ΕΠ μεταξύ Εικονικής, Επαυξημένης και Μεικτής Πραγματικότητας. Μειώνει επίσης το χρόνο ανάπτυξης ενώ αυξάνει τη διαθεσιμότητα των εφαρμογών ψηφιακής κληρονομιάς, ενθαρρύνοντας τους προγραμματιστές να τις διαθέτουν σε όλο το φάσμα ΕΠ, συμβάλλοντας έτσι στη διατήρηση της πολιτιστικής κληρονομιάς.

Attestation

I understand the nature of plagiarism, and I am aware of the University's policy on this.

I certify that this dissertation reports original work by me during my university project except for the following:

- The MAGES technology mentioned in Section 3.6 was primarily developed by my co-workers and belonged to the ORamaVR company.
- The 3D model of the museum, which I used for the development of the application of the Industrial Museum and Cultural Center of Thessaloniki, was given to me from the Ephorate of Modern Monuments of Western Macedonia and thus belonged to them.
- The Unity3D game engine entirely belongs to Unity. (https://www.unity.com)
- The UniZip plugin used for unpacking the ARKit SDK package belongs to its owner (https://github.com/tsubaki/UnityZip).
- ARCore SDK entirely belongs to Google.
- ARKit SDK entirely belongs to Apple.
- Oculus Integration SDK entirely belongs to Oculus.
- Mixed Reality Toolkit entirely belongs to Microsoft.

Signature *Efstratios Geronikolakis* **Date 14/09/2020**

Acknowledgments

I would like to thank my supervisor, Professor George Papagiannakis, for his support, inspiration, and providing the main idea of this tool, so that I could work towards it and make it a reality. Without him, I would have never been introduced to the unique and wonderful world of Computer Graphics.

I would also like to thank Michael Tsioumas and the Ephorate of Modern Monuments of Thessaloniki, for testing my digital heritage application, which was the main application on which the tool of this thesis was tested.

Special thanks to my co-worker, namely Paul Zikas, for being an excellent help for the writing of this thesis, as he always provided available research and academic work for me to study and include in this thesis, as well as Nikos Lydatakis, Steve Kateros, Michael Kentros and Stelios Georgiou for evaluating my work for this thesis. Also, I would like to express my gratitude to Amalia Kargopoulou for providing me with valuable comments for this thesis.

Finally, I would like to thank my family and especially my fiancée, Mary Varytimidou, for mentally supporting me in order to go through difficult and stressful phases of this thesis. I am always grateful for her support.

To my family

Table of Contents

List of Figures

List of Tables

1 Introduction

Throughout the years, computer graphics applications are becoming more and more prevalent each day. People use them for a variety of purposes, for instance, entertainment [55], education [19], training [56], even research [56]. Many are also used for the purpose of cultural heritage preservation [47].

A crucial term mainly mentioned in this work is XR. Extended Reality (XR) is a fusion of all the realities – including Augmented Reality (AR), Virtual Reality (VR), and Mixed Reality (MR) – which consists of technology-mediated experiences enabled via a wide spectrum of hardware and software, including sensory interfaces, applications, and infrastructures. XR is often referred to as immersive video content, enhanced media experiences, as well as interactive and multi-dimensional human experiences. [62]

"XR does not refer to any specific technology. It's a bucket for all of the realities."

-Jim Malcolm, Humaneyes

The number of digital heritage applications that exist across the XR spectrum is considerably limited. Discovering a Virtual Reality application, for instance, that exists in Augmented Reality as well, is a rare situation. It is comprehensible, considering that the two previously mentioned realities significantly differ in terms of user experience (interaction). As a result, the amount of effort needed for transitioning such an application from Virtual Reality to Augmented Reality or vice-versa (across the XR spectrum) would be approximately identical to that of developing it from scratch. This fact is the primary reason why developers do not consider releasing an application across the XR spectrum. It is quite unfortunate since there is always a chance for it to be more appealing in another reality than the one it was initially designed for.

Depending on the application content and its general purpose, it might be more suitable for Virtual Reality than Augmented Reality, for instance. Such an example would be training applications [51]. In such cases, we need the users to fully immerse themselves and try to remember the steps and movements they executed (utilizing the controllers) inside the virtual world. That would not be possible for a mobile device with Augmented Reality.

The previously mentioned problems would be addressed if the transitioning procedure across the XR spectrum was quick and could be executed effortlessly. Developers would quickly port their application through the XR spectrum and discover the most appropriate case for each

one. This problem can be addressed by developing a framework, able to be used through the game engine, which would automatically transit the application project across the XR spectrum (generating a ready-to-operate project) while respecting the developers' choices.

1.1 Scope and Objectives

This work aims to relieve the developers (and especially new ones) from a burden they regularly face: the transition of a computer graphics application across XR (from VR to AR, for instance) and more specifically, virtual museum ones. This transition usually requires many actions from the developers' side. Of course, tutorials that exist online only explain how to start an application in the chosen reality from scratch, as shown in [53]. Similar are the cases for the other platforms/realities as well. Our tool aims to relieve developers from this confusion and offer them a swift solution to their problem, which will be an operating project for the reality of their choice for their application.

Another intention of this work is to reveal how a digital heritage application can transcend across XR, (meaning adapting the needs of the application to the new reality/device requirements) by using our framework. It solves the problem above and is a part of the results of this work. An example of such application XR transition is described in [48], where an application transits from Virtual Reality to Mixed Reality manually. Our framework automates an essential part of this process.

1.2 Achievements

The significant achievement of this work is the presented framework, able to provide XR migration. We created a collection of scripts for use regarding this framework, which provides an "XR transition library" where developers can view and download the available SDK for their favorite platform. This collection of C# scripts also installs and sets up the environment during the platform switch, so developers will not have to remember or search all the extra needed actions. It is quite a useful framework, especially for new developers, but even for experienced ones, if they would like a fast platform set up, a switch in the target platform of their project, or starting a new one.

We had some other accomplishments, which were also elements of this work: a new digital heritage application and a port/refactoring.

To measure the efficiency of our system, we utilized an application we developed regarding the cultural heritage preservation of a museum in Thessaloniki. We initially created it for iOS mobile AR using Apple's ARKit [63], and by using our framework, it became available for Android mobile AR devices as well as for Android mobile VR (Oculus Go). Besides, another

achievement that led to the development of our XR transition framework was the manual platform switch of another digital heritage application in VR, rendering it available to operate on Microsoft HoloLens [48].

One of the main challenges we faced was the transition of the VR digital heritage application to Microsoft HoloLens [48]. As a result, another achievement of this work is the mechanism we created to simulate the interaction that the VR version of the application offered to the users. For instance, the two hands interaction system using HoloLens SDK, as well as the completion of specific action-driven events, which is a part of the current application SDK. This achievement inspired us to continue studying and researching to develop a framework (the one we present in this work) to provide our newest digital heritage application to many platforms in scarcer time.

1.3 Overview of Dissertation

We split this work into five sections. Broadly, this dissertation's main roadmap is the state-of-the-art, the information about the devices and their platforms, the definition of the problem, the challenges we faced, its solution and our framework's evaluation. We illustrate this work plan in more detail below.

In the second section, we mention some other inspiring digital heritage applications and some authoring tools that contribute to the development of such applications. Following that, we discuss some applications that exist across XR, which is what we designed our tool to do automatically. Then, we cite other XR applications, for which their primary purpose was to educate the users (serious games), and finally, we present our publications regarding this work.

In the third section, we introduce the essential elements of each of the platforms and devices used for this work. We also provide information about their origination and SDKs. Afterwards, we declare the definition of the principal problem of this work. Specifically, we define the problem, and then introduce our manual solution for it and the need to automate this process. Moreover, we present the digital heritage application that we also created as a part of this work. This application is a serious game regarding the Industrial Museum and Cultural Center of Thessaloniki [47] and exists across XR. Then, we provide some examples of transcending a single application across XR. Furthermore, we describe the case study of the digital heritage application of the museum in Thessaloniki. Then, we present our framework, the "XR Transition Manager", which performs a fast and easy transition across XR for an application.

In the fourth section, we present our framework's evaluation scheme, along with the results, and we draw some conclusions. Lastly, in the fifth section, we discuss the future work we could do, in order to improve our tool further.

2 State-of-The-Art

There are several types of work that we will present in this section. Most of them concern the preservation of cultural heritage because our framework is targeted towards the transition of virtual museum applications across XR. We will present some exciting state of the art virtual museum applications. As our framework will also involve the ability of transition across XR, we will examine some existing authoring tools and applications that work in various realities. Finally, we will momentarily report our previous work in this domain.

2.1 Virtual museums

Nowadays, virtual museums have become more and more prevalent. The idea of designing a virtual version of a real museum is a fascinating one, as it is a way to contribute to cultural heritage preservation. That is because, by generating a virtual museum application and distributing it to people, they can gain access to the virtual version of the particular museum from anywhere in the world. It is a very innovative idea since every person that maintains a mobile device will virtually visit the museum of their desire, without having to travel directly to it, making their life more convenient. We present some exciting work in virtual museums below.

2.1.1 Holographic Virtual Museums

Since the advancement of holographic technology, AR headsets are evolving, including interactive features like gesture and voice recognition and improvements on resolution and FOV. Besides, untethered AR headsets paved the way for mobile experiences without external processing power from a PC. Such embedded systems facilitate excellent tools to represent virtual museums [13] due to their lack of cables and enhanced interactive capabilities. Virtual Museums are institutional centers in society's service, open to the public for acquiring and exhibiting the tangible and intangible heritage of humanity for education, study, and enjoyment.

True Augmented Reality is another technological advancement that can benefit virtual museums due to its very realistic results. It has recently been defined as a modification of the user's perception of their surroundings that the user cannot detect [14] due to their realism. Virtual characters and objects blend with their surroundings, attaining the "suspension of disbelief".

Many approaches on holographic digital heritage applications emerged in modern years, each one concentrating on a different aspect of representing the holographic exhibits within the real environment. A published survey [15] investigated the impact of Virtual and Augmented Reality on museums' overall visitor experience, highlighting the social presence of AR

environments. [16] presented a correlation of the latest methods for the rapid reconstruction of real humans using as input RGB and RGB-D images. They also propose a complete pipeline to compose highly realistic reconstructions of virtual characters and digital asserts suitable for VR and AR applications. Storytelling, Presence, and Gamification are three critical fields that should be considered when creating an XR application for cultural heritage. [17] presented a comparison of existing MR methods for virtual museums and pointed out the importance of these three fields for applications that contribute to cultural heritage preservation [18]. Furthermore, in [2], fundamental elements for MR applications alongside examples are presented.

Another recent example [19] introduced two Mixed Reality Serious Games in VR and AR, comparing the two technologies over their capabilities and design principles. Both applications showcased Knossos's ancient palace in Minoan Crete, Greece, through interactive mini-games and a virtual/holographic tour of the archaeological site using Meta AR glasses. [20] successfully published an AR application for visualizing restored ancient artefacts based on an algorithm that addresses geometric constraints of fragments to rebuild the object from the available parts.

2.1.2 Survey of Recent MR methods for Virtual Museums

Table 1 below summarizes critical papers in the last nine years, after the previous relevant survey paper from [21] is presented. Although there is no specific MR method that features gamified storytelling with heightened interaction that still maintains full immersion and the feeling of presence, several conclusions and recommendations for the next research lines can be drawn and summarized.

The MR technologies used in the key papers (located in Table 1 below) contribute to preserving cultural heritage, each one with its level of storytelling, presence, gamification, interaction, and tracking methods. As all the installations below are MR applications, many of them consider the gamification field to be exciting and fun for the viewers. Some of them include storytelling components (e.g., Papefthymiou et al. [22], Pedersen et al. [23]), which are utilized to inform the viewers about the story of a monument, for instance, thus contributing in cultural heritage curation. Furthermore, most MR methods below support partial immersion, while a few support full immersion. The term “immersion” is deliberately included because it can be easily quantified based on the display, whereas “presence” is elusive and depends on many parameters and thus, it is challenging to provide that it exists throughout a simulation. VR Head-Mounted Displays (HMDs) thus support full immersion, whereas AR and holographic AR support partial immersion. Moreover, applications that run tethered with a computer and do not support VR or AR provided no immersion. The majority of these papers use mobile AR or holographic AR, which explains most of the partial immersion entries in

Table 1. A recommendation at this point would be to develop more MR applications that support desktop or mobile VR with full immersion, as it creates a more true-to-life experience for the viewers. The feeling of presence is respectably higher than it is with partial immersion. These technologies can also be categorized into two additional categories, tethered and untethered, depending on whether an installation needs a connection to a computer or not (standalone device). An illustration of an untethered MR technology, in the table below, is the Papaefthymiou et al.[22], which uses the Apple Ipad Pro device in order to operate. It uses Apple's ARKit for camera tracking. Cables do not limit the viewers' movements (since it is untethered), enhancing the feeling of presence as their movements would not be limited, feeling as if they were exploring a real archaeological site (in this case). Besides, it supports gamification and storytelling elements along with the feeling of presence and freedom of movements. All these elements create a flawless experience for the viewers. Another version of this work operates on the Microsoft HoloLens Holographic AR HMD, which is also an example of untethered AR.

Mixed-Reality method	MR Installation	Gamification	Storytelling	Interaction	Tracking	Immersion	Intangible Heritage
Anderson et al.[21]	Rome Reborn Serious Game	x	x			-	No
	Ancient Pompeii application	x	x			-	No
	Parthenon Project	x	x			-	No
	Virtual Egyptian Temple	x	x	Walking/movement	Cave Automatic Virtual Environment (CAVE)	Full	No
	The Ancient Olympic Games	x	x	Navigation wand of the VR system		-	No
	Virtual Priory Undercroft	x	x			-	No
	Commercial Historical Games	x	x			-	No
Bugalia et al.[31]	3D printed models, projector, camera, laser pointer	x	x	Laser pointer	IR camera, project camera, view camera	Partial	No
Dong et al.[32]	VR application using Focus3D X 330 Faro Scanner, Spheron PanoCam, two GoPro Hero 4 cameras for scanning and Unity for rendering	x		Depending on VR platform	Depending on VR platform	Full	No
Drossis et al.[24]	3D model prototype (depth sensor, touch screen, interactive cube, projection)	x		Touchscreen, interactive cube, walking / movement	RGB-D sensor, Kinect or Asus Xtion camera	Partial	No
Gimeno et al.[33]	Region of Valencia map, 7 pointers and hosts	x		Walking / movement	Map, pointers, Kinect camera	Partial	No
Grammenos et al.[34]	Tabletop augmented reality system		x	Finger-based input	Projector and pieces of white paper	Partial	Yes/No
Javornik et al.[35]	Apple Ipad Pro	x		Touchscreen	Inner iPad camera	Partial	Yes

Kitsikidis et al.[36]	Microsoft Kinect sensor	x		Motion Capture	Skeleton tracking with Kinect sensor	Partial	Yes
Kosmalla et al.[37]	HTC Vive with Leap Motion	x		Controllers	Positional and hand tracking	Full	No
Koutsabasis et al.[38]	Leap Motion	x		Hand ges-tures	Hand and finger tracking	Partial	No
Liarokapis et al.[40]	HTC Vive	x		Controllers	Motion and laser based	Full	No
Margetis et al.[40]	Interactive maps tabletop system		x	Handwrit-ing, gestures, touch/click	Projector, high resolution camera, depth sensor	Partial	Yes/No
Nakevska et al.[41]	CAVE	x	x	Back-projection on walls	Pressure sensors	Full	Yes/No
Papaefthymiou et al.[22]	Oculus Rift	x	x	Controllers	Rotational and Posi-tional (Sensors)	Full	No
Papaefthymiou et al.[22]	Apple Ipad Pro	x	x	Touch Screen	Apple's ARKit for camera tracking	Partial	No
Papaefthymiou et al.[22]	Microsoft Ho-lolens	x	x	Voice Commands, Gestures	Body Motion	Partial	No
Papagiannakis et al.[42]	DELL P4 M50 Mobile Workstation	x			Real-time markerless camera tracking	Full	No
Pedersen et al.[23]	Meta developer kit	x	x	Gestures	Markerless surface tracking, head movement	Partial	No
SpatialStories[26]	Toolkit for VR/AR Platforms	x	x	Depending on VR/AR platform	Depending on VR/AR platform	Full / Partial depending on the plat-form	No
Tisserand et al.[25]	Computer with Kinect sensor for Traditional Sports Preservation	x		Movement	Kinect sensor	Partial	Yes

Table 1. Comparison of recent MR methods for virtual museums. [17]

On the other hand, there are exceeding MR applications (e.g., Pedersen et al. [23], Drossis et al.[24], Tisserand et al.[25]), supporting gamification and storytelling elements, though they limit viewers' movements due to the existence of cables or the need for a connection with a desktop PC.

Although these applications are impressive and significantly contribute to cultural heritage preservation, they restrict the users' freedom of movement, which in some cases may disrupt the feeling of presence. Based on this analysis, it is recommended for MR applications to operate in MR installations that do not restrain the movements of the viewers in any way. Without the restriction of movements and utilizing full immersion, a viewer's experience will reach very high levels. Finally, SpatialStories [26], a very modern commercial effort, is a toolset for real-time interactive VR/AR experiences featuring storytelling for non-programmers. Although it is not thoroughly tested, it poses a promising commercial solution in contributing

to MR digital heritage applications from information gathered from their website and its videos.

2.2 Authoring tools for content creation in MR

Content is an essential part of a 3D application (if not the most crucial one). Extensive care must be taken during content development, as it constitutes the application and user experience base. Since creating content is a time-consuming and challenging procedure, different authoring tools for content creation have come to the surface to ease developers' lives. In this section, we present some of the prevailing works in this field.

2.2.1 Platforms for Gamified Content Creation

Authoring tools and additional content creation platforms emerged in modern years to fulfill the demand for interactive MR applications. BricklAyeR [27] is a collaborative platform designed for users with limited programming skills that allows the creation of Intelligent Environments within a building-block interface. ExProtoVAR [28] is a lightweight tool to produce interactive virtual prototypes of AR applications designed for non-programmers lacking AR interfaces experience. RadEd [29] highlights a new web-based teaching framework with an integrated smart editor to create case-based exercises for image interaction, such as taking measurements, attaching labels, and selecting specific parts of the image. It facilitates a framework as an additional tool in complex training courses like radiology. ARTIST [30] is a platform, which provides methods and tools for real-time interaction between human and non-human characters to generate reusable, low cost, and optimized MR experiences. It aims to develop a code-free system to deploy and implement MR content while using data from heterogeneous resources semantically. The aforementioned solutions provide developing environments to generate MR experiences. However, they lack advanced authoring tools and educational curriculum to support advanced educational - training scenarios. Lastly, in [50], the authors propose a gamified way of content creation for a training application, through a user interface by connecting blocks of events or setting up the desired events through Virtual Reality.

2.2.2 Unity MARS

Unity MARS [54] is a novel AR authoring tool, produced by Unity3D in 2020. It is unique since it offers the ability to develop and test AR applications from the Unity MARS environment without building the application each time. With the use of proxies, which are 3D objects imitating real-world objects, and "fuzzy authoring" developers can define the minimum and maximum measurements for them rather than code precise values. Moreover, with a relatively

simple drag-and-drop feature, developers can place their 3D models in the scene, and Unity MARS produces all the appropriate proxies and conditions for them. It also supports different kinds of real-world data, such as images and surfaces. In the near future, it will also support body tracking. It is a fascinating and time-saving authoring tool for AR that aims to speed up AR development.

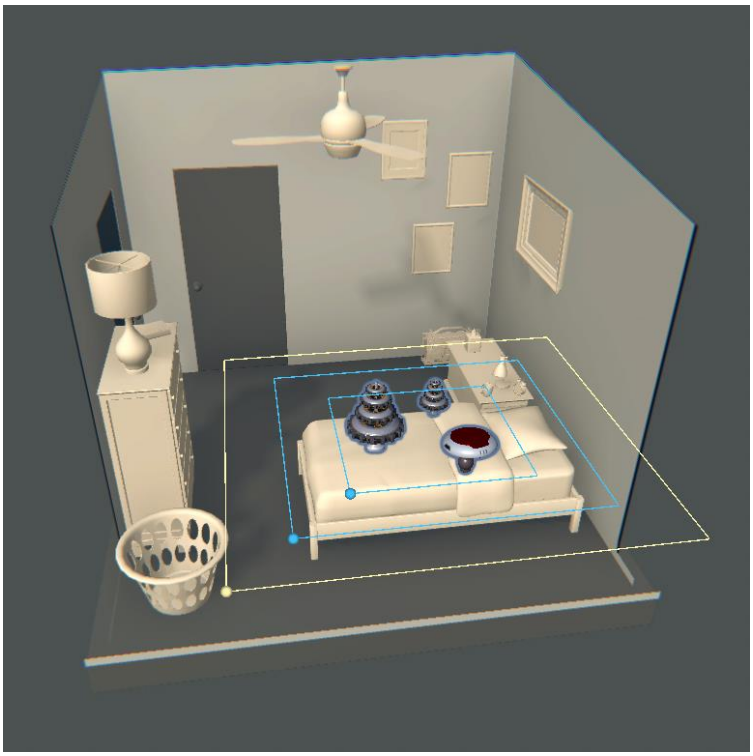

Figure 1. Example of the "Fuzzy Authoring" feature of Unity MARS.

2.3 Applications existing across XR

Following a rather extensive search for previous works on applications existing across XR, we did not find any scientific results. It is acceptable to the perspective that it is rather challenging for an application to exist across XR. From our experience, we believe that one of the reasons for this absence of examples could be the porting complexity. Of course, it is a procedure, which is not unachievable. It is doable, but since it is a time-consuming procedure [57], many developers refrain from commencing it. They prefer generating new content for a particular platform and push on. They do not ponder porting the same applications to other platforms, since they may consider this “recycling” of the same application. All the previously mentioned thoughts are based on our opinion, of course.

Another reason could be that apart from the challenge of the porting procedure, each platform has its specifications and requirements. As a result, an application in Virtual Reality that uses controllers would need total rework to operate in a mobile device and Augmented Reality. Mobile devices do not encourage the use of controllers. Their primary input device consists of

a touchscreen. This fact completely changes the whole user experience, and it is reasonable to refrain the developers from attempting the port.

Multiple applications have certain principles and require a specific structure in order to offer their full experience to users. Suppose one of these principles is absent (in our example, the fact that users should be able to move their hands around freely and see them in the application). In that case, the immersion and, as a result, the user experience drops significantly.

We managed to spot one example, nevertheless. That would be a Virtual Reality application, which exists in Mixed Reality as well (HoloLens). Within this application, users perform a surgical operation, named Total Knee Arthroplasty. More specifically, users follow a sequence of actions, visualized by holograms, to complete the operation. We found out that an approach to transfer this application to HoloLens exists. However, it remained incomplete (it does not contain the whole operation), probably because of the reasons we explained.

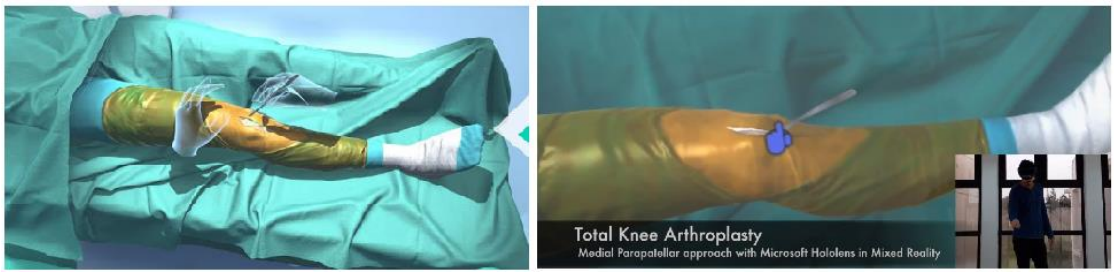

Figure 2. Total Knee Arthroplasty in Virtual Reality (left) and Mixed Reality (right).

2.4 Mixed reality serious games for smart education

Considering we consumed a large number of resources to produce our virtual museum application (for the museum in Thessaloniki) and worked to make it available for multiple platforms, we scrutinized for work regarding Mixed Reality serious games. We performed this research to determine the gamification methods that were used by researchers in their respective works, to collect ideas for our virtual museum.

In [43], the authors designed an application related to Knossos archaeological site. This application intends to educate users concerning the history of Knossos through a series of minigames. They created several versions of this application across XR. They perceived that the same application could not be applied across XR because each one had its boundaries and characteristics. In [44], they clearly define the meaning and significance of serious games and gamification while presenting different technologies for immersive heritage applications. Another work, bearing valuable information for constructing Mixed Reality applications for cultural heritage, is specified in [45]. Similarly, in [46], the authors present an inspiring, serious game application for intangible cultural heritage and, more explicitly, dancing. Users can

learn many traditional dances by following a sequence of movements, while the software provides them with feedback regarding how well they performed these movements.

2.5 Our publications related to this work

Apart from the current project, we participated in some other exciting works and studies as well. A brief overview of these works follows:

- **Rapid Reconstruction and Simulation of Real Characters in Mixed Reality Environments [16]:** In this study, we compare several 3D reconstruction methods for real characters—namely, Agisoft Photoscan Software, Fast Avatar Capture Application, and Occipital Structure Sensor. The outcome was that the reconstruction through Occipital Structure Sensor yielded the best results. That is because users can capture data from any point of view and distance around the subject, thus delivering high-quality textures and more solid geometry.

Figure 3. Digitization of the priest of the Asinou church, using the Occipital Structure Sensor.

- **New Cross/Augmented Reality Experiences for the Virtual Museums of the Future [47]:** We submitted this work to the Euromed 2018 conference. Here, we present our first version of the cross-reality application for the Industrial Museum and Cultural Center of Thessaloniki. We showcased an early version of the application running on iPad Pro 2017 and the first draft of its Android version. We presented the application in detail and analyzed each of its functions.

Figure 4. Screenshots from the Industrial Museum and Cultural Heritage of Thessaloniki cross-reality application.

- **Mixed Reality, Gamified Presence, and Storytelling for Virtual Museums [17]:** In this study, we presented and compared different approaches of applications (and work in general) regarding virtual museums concerning the preservation of cultural heritage. This rather extensive study provided beneficial results for researchers and readers.

- **A True AR Authoring Tool for Interactive Virtual Museums [48]:** This study comprises the most relative work for this project. In this work, we present the steps and actions needed to transfer a Virtual Reality application (operating on desktop Oculus Rift) to Mixed Reality and Microsoft HoloLens. It inspired us to proceed further and devise the framework, which will be presented later in detail.

Figure 5. The user is performing actions, which were previously performed with the use of controllers, with his hands in HoloLens.

- **From Readership to Usership and Education, Entertainment, Consumption to Valuation: Embodiment and Aesthetic Experience in Literature-based MR Presence [49]:** In this work, we examine how literary transportation further amplifies presence and affects user response vis-à-vis virtual heritage, by focusing on embodiment and aesthetic experience. It is a more theoretical work that renders useful results regarding a Virtual Museum model expressly suited to cultural heritage.

- **MAGES 3.0: Tying the knot of medical VR [51]:** In this work, we present MAGES 3.0, a novel Virtual Reality (VR)-based authoring SDK platform for accelerated surgical training and assessment. The MAGES Software Development Kit (SDK) allows code-free prototyping of any VR psychomotor simulation of medical operations by medical professionals, who urgently need a tool to solve the issue of outdated medical training. Our platform encapsulates the following novel algorithmic techniques: a) collaborative networking layer with Geometric Algebra (GA) interpolation engine, b) supervised machine learning analytics module for real-time recommendations and user profiling, c) GA deformable cutting and tearing algorithm, d) on-the-go configurable soft body simulation for deformable surfaces.

Figure 6. A cooperative REBOA training scenario mode where users can visualize the patient's arteries.

3 Our Methodology – Contribution

3.1 Basic XR Application Integration Elements

Each XR application has different essential elements regarding the camera and user interaction functionality. These elements vary depending on the platform and hardware that the application operates. The platforms/operating systems that most 3D applications use (in Unity3D), and we studied in this work are four. The first one is “PC, Mac & Linux Standalone”, for applications that operate on a desktop computer. The second one is the “Android” platform for applications that operate on a mobile device or headset that supports android. Next is “iOS” for mobile devices that run Apple’s iOS and, lastly, “Universal Windows Platform” for mobile devices utilizing Windows. The hardware on which most 3D applications operate is either VR Head Mounted Displays (HMDs) or mobile devices (smartphones, tablets, and more).

3.2 Virtual Reality Applications

Virtual Reality technology generates an artificial environment that immerses the users, causing them to believe that they are a part of it, doing there and being there. Although this technology has emerged long ago, it is in the latest years that it made its first steps in the market and became well-known. For the users to enter the virtual world that a VR application provides, they have to wear a VR head-mounted display (HMD), which precludes them from having access to the real world, as long as they wear it. As a result, users experience full immersion. This technology is used widely for many purposes, from entertainment [55] to training [56] and research [56]. In the following subsections, we will present some well-known HMD devices alongside their essential elements in Unity3D, needed to create an application that will operate on each one flawlessly.

3.2.1 Oculus Rift

The first version of Oculus Rift to be shipped for development was the Development Kit 1, which came out on March 29th, 2013. Oculus created another version of their headset for development, naming it Development Kit 2, which came out in July 2014. It was an upgraded version of Development Kit 1, featuring better resolution, higher refresh rate, positional tracking, a detachable cable, and the omission of the external control box's need. [52]

3.2.2 Oculus Go

Oculus Go is a relatively new device, which came out on May 1st, 2018. It is a Virtual Reality device, the first untethered HMD without any cables. Without tangible elements, users immerse into the virtual world with even fewer real-world interruptions, unlike the tethered version of VR.

The device's main drawback is that only three degrees of freedom are supported (3-DOF) due to the lack of a camera tracking system. Having noted that, users could only rotate in the virtual world. The device handles no information about the depth of the user in the real world. In other words, walking in the real world will not affect the 3D application that Oculus Go is running [59]. Recently, Facebook announced the discontinuation of Oculus Go to focus more on the next generation and Oculus Quest.

3.2.3 Oculus Quest

Oculus Quest is one of the latest devices that Oculus brought to the world. It came out on May 21st, 2019. It can be considered an upgraded version of Oculus Go since it is untethered and provides better hardware, two controllers, and six degrees of freedom (6-DOF). 6-DOF means that the device handles information about the user's depth. The system relies on four wide-angle cameras located on each corner of the headset to track the headset spatially through a SLAM system. [4]

3.3 Augmented Reality Applications

Augmented Reality technology merges the real with the virtual world. Applications made utilizing this technology usually operate on mobile devices containing at least one camera component. A camera is profoundly needed because this is the users' "window" to the virtual world, where the virtual 3D objects will reside. Augmented Reality offers partial immersion since users still have access to the real world. It is a beneficial technology, applied in many different situations, from entertainment [55] and education [55] to business [56]. Both iOS and Android mobile devices are eligible for operating Augmented Reality applications. Each platform has produced its version of Augmented Reality Software Development Kit (SDK), namely ARKit and ARCore.

3.3.1 Google's ARCore

Google's ARCore is an SDK for Augmented Reality applications that operates on Android. It was released on March 1st, 2018. The most recent Android devices [61] utilize it to create Augmented Reality experiences that blend with the digital and physical worlds. Conversely,

ARCore uses the Inertial Measurement Unit (IMU) to track and interpret data. It also measures the shape, the build, and the surrounding objects' features to detect and identify the right position and orientation of the Android device in use [5]. It supports light estimation by collecting data from the device camera to estimate the environmental light direction to enlighten the virtual objects in the AR scene.

3.3.2 Apple's ARKit

Apple's ARKit is an SDK for Augmented Reality applications operating on iOS. It was released in June 2017. It supports many of Apple's mobile devices and is used to generate Augmented Reality experiences for various purposes. ARKit uses a Visual Inertial Odometer (VIO) to achieve Motion Tracking in order to accurately track a position with respect to objects in the real world [5]. Moreover, devices equipped with ARKit can seize and process the surrounding environment (horizontal distances) – a function called Environmental Understanding. Subsequently, Light Estimation allows the cameras of iOS devices to detect real-world area light sources and light the Augmented Reality objects accordingly [5].

3.4 Holographic Augmented Reality

Holographic Augmented Reality, as the name implies, is very close to straightforward Augmented Reality. It differs in the way that all virtual objects are rendered as holograms. With Holographic Augmented Reality's aid, users can observe these holograms through a special HMD that highly increases realism. As a result, not only do they experience the presence of virtual objects being in their room, but they can also interact with them using hand gestures. Thus, Holographic Augmented Reality is bound to partial immersion, since users have access to the real world, but in this case, realism is even higher compared to straightforward Augmented Reality.

3.4.1 Microsoft HoloLens

HoloLens is a Head Mounted Display unit connected to an adjustable, cushioned inner headband that can tilt HoloLens up and down, forward and backward [6]. To wear the unit, users must adjust the wheel at the back of the headband to secure it around the crown, supporting and distributing the weight of the unit equally for comfort [7], before tilting the visor towards the front of the eyes [6].

The unit front houses many sensors and related hardware, including processors, cameras, and projection lenses. The visor is tinted [7]; enclosed in the visor piece is a pair of transparent combiner lenses, in which the projected images are displayed in the lower half [8]. HoloLens

must be calibrated to the interpupillary distance (IPD) or accustomed vision of the user [9][10].

Accompanying the bottom edges of the side, located near the user's ears, are small, red 3D audio speakers. Competing against typical sound systems, the speakers do not obstruct external sounds, allowing users to hear virtual sounds and the environment ones [7]. Using head-related transfer functions, HoloLens generates binaural audio, which simulates spatial effects, allowing users to virtually perceive and locate a sound, as it is originating from a virtual pinpoint or location [11][12].

3.5 Reality Transition Methodology

One of the main challenges our system resolves is the porting of immersive applications across the realities spectrum. We will begin by presenting the porting methodology of an application initially generated for Virtual Reality (Oculus Rift) to operate on Microsoft HoloLens.

3.6 The Virtual Reality Digital Heritage Application

Our Sample app [56] is an MR application in which we present users with basic examples of all the functionalities M.A.G.E.S SDK supports. It is a playground for MR. We consider it a room where users can experiment with M.A.G.E.S SDK's basic mechanics, try them, and even create their own, using our tools. These mechanics can be applied to other scenes, as they are not bound only on this specific application. Users can experience simple examples of various mechanics and interact with many objects in the scene (pick them up, hold them, even throw them), thanks to our “interactable item” utility that M.A.G.E.S SDK provides. They can utilize this functionality to potential objects in order to interact with them and move them around the scene with their virtual hands.

The virtual hands are another interesting mechanic that M.A.G.E.S SDK contains. We set them up automatically to follow and respond depending on the controllers (Oculus, Vive, Mixed Reality, and more). Users can effortlessly set these hands to interact with objects of their decision. The hands are animated to perform the appropriate real-life gesture. This functionality increases the realism and thus the feeling of presence in the scene.

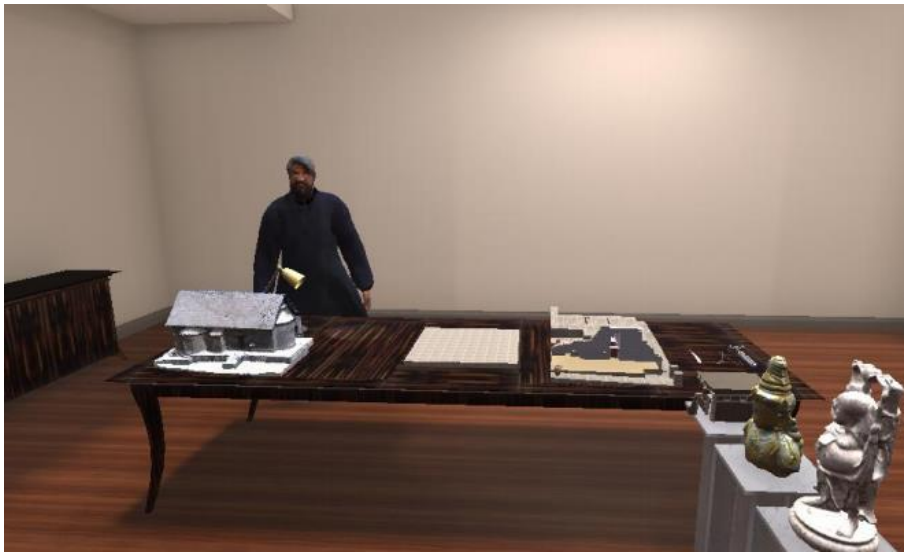

Figure 7. Our sample Virtual Reality application

3.7 Definition of the problem

For a better perception of the current situation, we have to determine the main issue. Thus, we divide it into two sub-problems: “Immersion and surroundings” and “Interaction”. Then, we try to provide a solution to those, thus solving the root one.

3.7.1 Immersion and surroundings

This task contained two significant challenges. Firstly, we had to prepare an application to operate on a completely different reality. The VR environment is somewhat different from the holographic that HoloLens supports. In VR, users do not have access to the real world. They fully immerse themselves in the virtual world. Such is not the case with holographic AR, where users have access to the real world. While wearing the HoloLens device, they see augmented holograms (3D objects, which are not fully opaque) occupying the real world. These holograms are in the same room with them, and users can observe them as a result.

Considering that users utilizing HoloLens have access to the real world, there is no need for their view to be surrounded by virtual obstacles, as in VR. The main goal of HoloLens is to augment the users’ world and not to set barriers in it. For that reason, it is characterized by partial immersion, in contrast with VR and its full immersion.

3.7.2 Interaction

The second and most exciting challenge we faced was to find a way to support the interaction that VR offered, with HoloLens. There is a significant gap between these two devices, apart from the whole reality and immersion difference described before. Oculus Rift provides users with interaction through its two controllers. With their help (one controller for each hand), the

application simulates the users' hands. By utilizing them, they can interact with virtual objects. By pressing the controllers' buttons, they can observe their virtual hand moving and picking up objects, simulating the movements that their real hand would do in these situations.

Since HoloLens does not support any controllers, we had to think of a way to simulate such interactions. One great advantage that HoloLens incorporates is its hand tracking system. This mechanic recognizes the position of the users' hands and some specific hand poses. We had to find a way to simulate all interactions that VR offered, utilizing the users' hands, and taking advantage of the specific hand gestures that HoloLens recognizes. Additionally, we thought that by using the users' real hands to interact with virtual objects, the realism would increase even more in comparison with VR.

3.8 Elaboration

Having defined the problem above, we decided to begin searching for a solution. Since we did not find a way to automate the procedure, we realized we had to start developing through a manual way to overcome this challenge.

3.8.1 Setting Up the Universal Windows Platform

During the platform switch procedure, we had to define some settings. Unity3D supports many platforms. The selected one must be that on which the current project will operate. While determining a platform, one must know the project target device and the target device's operating system. For the case of HoloLens, we select the Universal Windows Platform, as its operating system is Windows 10.

Apart from the platform selection, other settings need to be defined in order for our application to operate correctly on a HoloLens device. We need to have the Mixed Reality SDK installed in our Unity3D project, as the HoloLens camera object needs some scripts to operate correctly, which are a part of this SDK. Furthermore, it is needed to support HoloLens interactions. There are many versions of this SDK. We can receive it by browsing the web and downloading the version of our choice. After downloading and installing this SDK, we should change the camera background from "Skybox" to "Solid Color" because, as we mentioned above, in holographic AR, users should not be surrounded by virtual obstacles.

On the contrary, the virtual world should blend with the real world harmonically. Besides, we must add the suitable scripts and object hierarchy to the camera; otherwise, we must use the appropriate camera prefab that the SDK provides. We must also set the quality level of graphics for HoloLens to "Very Low", as it does not have much processing power at its current

state to support ultra-high quality graphics. Moreover, since the 3D models are holograms and not fully opaque, some details would not be visible either way.

Before initializing the build procedure, for a sample HoloLens application, we have to set some preferences in the “Player Settings” of the current project. Specifically, for HoloLens, we need to go to “Player Settings” and locate the “Capabilities” section, found in “Other Settings”. All HoloLens applications must have “SpatialPerception” enabled. We also enabled “InternetClient” and “Microphone” as our application contained a login identification mechanism that utilized the internet. In some cases, we used the device microphone to give voice commands.

Eventually, having set the above, we must set our target device to be HoloLens, through the build settings menu. The application is ready to be built.

3.8.2 Interaction support

To support interaction through HoloLens, we had to integrate the interaction system of HoloToolKit with NewtonVR, which we used in our VR project. Currently, the most common way to interact with holographic objects through HoloLens is through the pinch gesture to grab a hologram and change its position within the virtual environment. This method utilizes a simple parenting mechanic to grab the object just by switching its parenting to be the user’s hand position. This mechanic is simple but offers restricted functionality and weak user experience. To improve the parenting grab mechanic and unify the interaction mechanic in our platform, we integrated the HoloLens gesture grabbing system to the NewtonVR system. The diagram below illustrates the interaction module, which handles the input from HoloLens and forwards feedback to the platform.

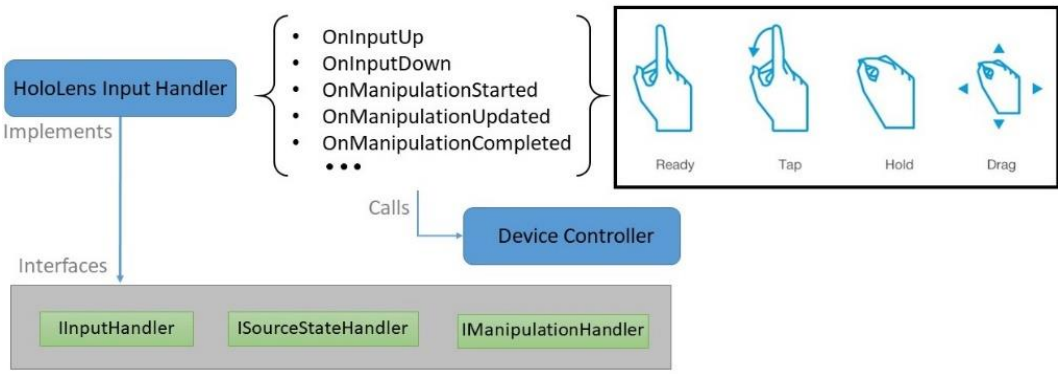

Figure 8. The architecture of our HoloLens input handler.

Our methodology was the following. We needed to generate an intermediate module between the device controller and the HoloToolKit. This module is called the HoloLens Input Handler. It implements three interfaces from HoloToolKit to link the inputs from HoloLens gestures

and vocal controllers directly to our application. For instance, we call the OnInputUp method when users raise their pointer, indicating the first stage of tapping gesture. When the HoloLens Input Handler recognizes this gesture, it automatically calls the appropriate Device Controller method to signal our application for a possible gesture performance.

3.8.2.1 Porting a Virtual Reality application to an Augmented Reality system

At this point, we implemented the interaction module to handle virtual assets with natural gestures. The next action is to reconstruct the augmented environment, where users will interact. We initially designed our application for a VR environment. Thus, the 3D assets visualization and the virtual room were entirely digital. However, in AR applications, the rendered environment blends with the virtual and the real world (as the augmentations do not cover the entire Field of View). Nonetheless, they are placed in critical locations respecting physical objects.

More particularly, to design the AR application, we only have to keep a small number of digital assets and delete the majority of them to create room for the real environment. Thus, we only kept the wooden table from our 3D room, alongside the “interactable items” and the priest of Asinou. Also, to improve the holographic assets' realism, we integrated a shadow plane under each object to replicate a real-time shadow. This technique is simple enough, yet it enhances the field's depth, including an additional layer of illumination.

Another module we need to consider when switching the deployed medium (AR/VR) is the camera object, representing the HMD. For this reason, we integrated the HoloLens camera from the HoloToolKit into our system to support both cameras and technologies. In this way, developers can set the camera with a single click without importing any additional packages or libraries, transforming the sample app into a plug and play system.

3.9 The application porting result

The outcome was immensely gratifying. We succeeded in porting our VR application on HoloLens. The feeling was very realistic since we could pick up objects with our real hands (instead of controllers). This level of realism made the application even more delightful.

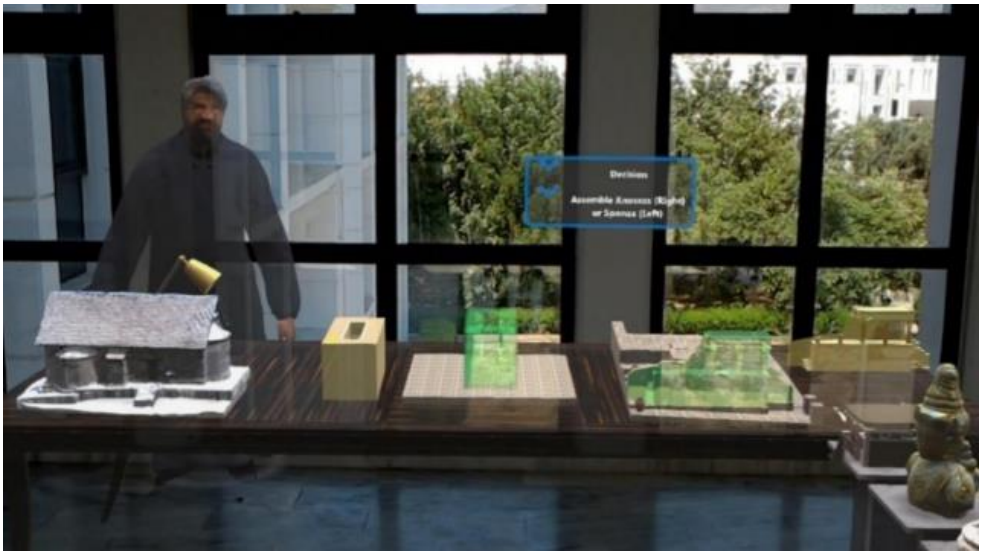

Figure 9. Our sample application while running on HoloLens.

Afterwards, we realized the enormous amount of time we devoted to this work. We deemed that it would be better if there was a way to automate all these steps and procedures. Remarkably, for someone who had not used HoloLens before, it would be a highly time-consuming procedure to discover and download the latest version of HoloToolkit, set up the project settings correctly, and place the appropriate objects in the Unity3D scene [48]. It is what steered us towards the development of the "XR Transition Manager".

3.10 The Virtual Museum application

Each newly developed technology needs a test environment. This environment should be ideal for the specific technology's needs and should be used to test the emerging technology to its maximum. For our framework, we will utilize a highly particular application for testing. It is an application [47] that we produced voluntarily in order to contribute to the preservation of cultural heritage. It is about the Industrial Museum and Cultural Center of Thessaloniki. The initial porting of this application was between different SDKs, while being in the same reality (from ARKit to ARCore). Later, we performed a manual transition from Augmented to Virtual Reality first and tested that transition with our framework as soon as it was finished. The application, as well as the challenges we faced during its manual XR transition, are presented below.

3.11 The Industrial Museum and Cultural Center of Thessaloniki

The building of the Industrial Museum and Cultural Center in Thessaloniki has a fascinating history. It is the only remaining structure of the "Hamidie" complex, which was founded during the city's last ottoman period (1875) as an Orphanage ("Islahane") and a School of Arts and Crafts. The complex is located in the region of Evangelistria on the eastern side of the city

walls, both within and outside the historical city limits. After the liberation of Thessaloniki, the building's ownership passed to the Greek state, which rented out the complex from 1920 onwards to accommodate usage commensurate with the workshops that were initially housed in the School [13].

In 1992, the Ministry of Culture and Sports designated the building complex and its equipment as a listed historical monument. In 2011, the project "Restoration of the listed complex of the former School of Arts and Handicrafts ("Hamidie School") and conversion into an Industrial Museum and Cultural Center" was included in the Operational Programme of Macedonia and Thrace as part of the NSRF 2007 - 2013. The building complex's restoration and reuse were completed in 2015 thanks to the Greek state and the European Union's funding.

Figure 10. The Industrial Museum and Cultural Center of Thessaloniki.

3.12 A Cross-Reality Experience

The application development began in early 2018. In cooperation with the Ephorate of Antiquities, which kindly provided us the museum's 3D model, we created an application through which the users can virtually visit it and explore it as they would do in real life. They can enter the virtual museum through a portal that they place in the real world.

3.12.1 Crossing realities through a portal

As mentioned above, the virtual world's entrance to the Industrial Museum and Cultural Center of Thessaloniki is available through a portal. When the application starts, it enables the camera of the device. Users are then prompted to touch a flat surface (highlighted to help the user detect it quickly) to place the portal on it. After they place the portal, they can walk through it and enter the museum's world. As they enter, they can still look back, through the portal, to observe the real world (the area through which they came in). If they walk through the portal again, they will return to the real world, as their device's camera will show their real surroundings instead of those of the virtual world. The portal's addition increases the realism of the application and the feeling of XR transition (real world to the virtual world and vice versa). Since this application is available across XR, to test this work, the portal tool will be

absent in Virtual Reality since it provides full immersion, prohibiting the user from having access to the real world.

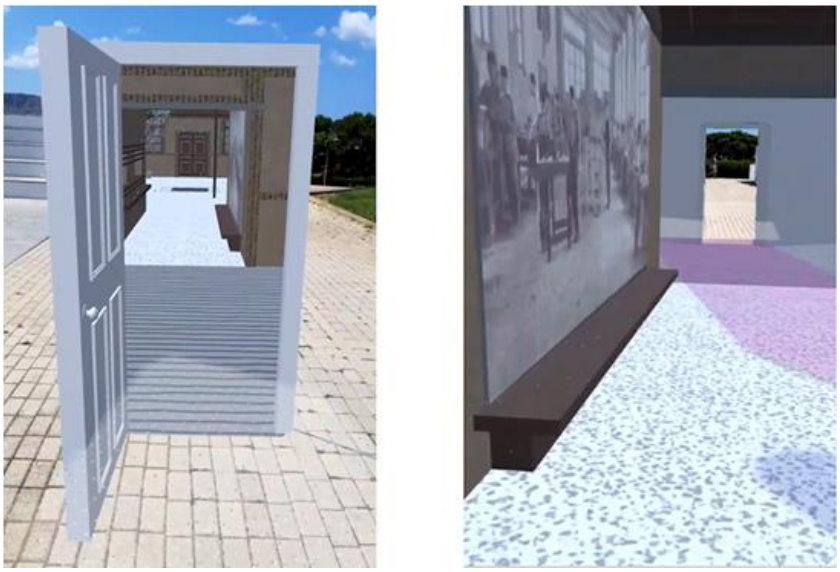

Figure 11. The portal leading to the cross-reality museum (left) and the view of the portal while inside the museum (right), during the early development stages of the application.

3.12.2 Essential elements of the digital heritage application

This application is a virtual museum incarnated. It attempts to combine the three fundamental fields of an XR application for cultural heritage, namely storytelling, presence, and gamification.

3.12.2.1 Storytelling field

The purpose of storytelling field in virtual museum applications is to inform users about the museum's story in general, or more specifically about the artifact that they are observing. For successfully applying this field to the application, we placed several information points around the virtual museum. When users approach one of these, they reveal information about the exhibit they are currently observing.

3.12.2.2 Presence field

‘Presence’ refers to the phenomenon of people behaving and feeling as if they “are there” in the virtual world created by computer displays [2]. The feeling of presence resides in our application through the meaning of cross-reality. Users walk through the portal, while the real-world transforms through their camera into the museum. Their real-world movements are mirrored in the virtual one, implying that if they move forward, they perform the same movement in the application. To explore the virtual museum, they have to move in the real world, which

positively enhances realism and the feeling of presence through partial immersion. They still have access to the real world when looking away from the screen of their mobile device.

3.12.2.3 Gamified content

A significant objective of gamification is the improvement of user's engagement [3]. Keeping users' interest and not letting them get bored and probably quitting the application early is crucial to the gamification field and needs attention. Gamification elements are also applied since it includes a quiz mini-game for users to test their knowledge regarding the museum and its exhibits. After reading/studying the information displayed at the information points, users can enter the virtual museum's unique "quiz room" and answer relative questions. Depending on the result, they obtain a virtual certification of their knowledge. They can also get "locked out" of the certification acquisition, but can always try again for a better score, providing the application with an immense replay value.

3.13 Significant challenges faced during the need of the first port

We created the application exclusively for iOS, and it was operating on an iPad Pro 2017. We displayed it at the museum curators, who showed much interest in it. This application's future goal is to publish it on an online store (like "App Store", for instance), for the public to download and use it. Since iPads are considerably costly devices, keeping the software exclusively for iOS would prohibit it from being downloaded by more people. Android devices, on the contrary, cover all price ranges, rendering them relatively affordable for the masses. It is where the need for the Android port arose. Nevertheless, we needed two significant variables to consider for this port: performance and the camera, which we describe below.

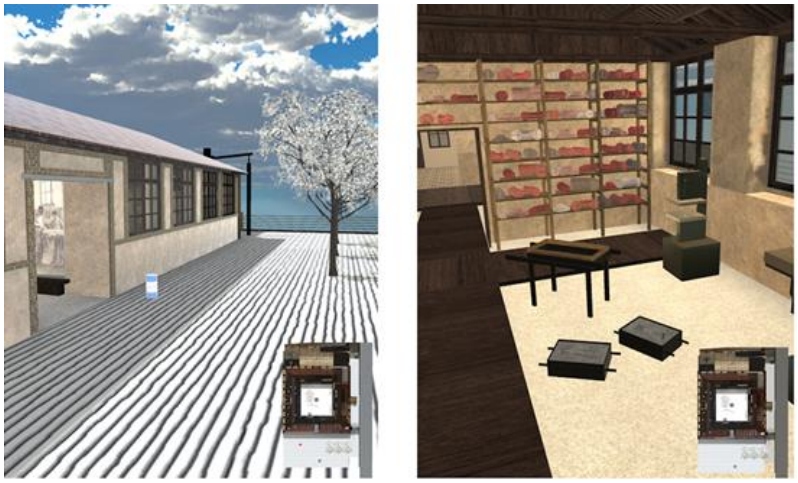

Figure 12. Standing outside the cross-reality museum (left) and being inside one of the museum rooms (right). In both images, we can see the mini-map in the lower right corner, displaying the position and rotation of the user.

3.13.1 Script performance

Performance was a significant challenge we encountered. Unfortunately, in the beginning, we did not give enough attention to the application performance, as the iPad Pro we used as a test device already had compelling specifications, causing our application to operate all the time smoothly. The same did not hold for every Android device, however.

This issue's primary reason is that we did not consider some complex components that we were not using, but were still in the scene, enabled. Of course, we could not have known about them earlier. As we previously mentioned, the iPad's high specifications did not allow us to understand that something was wrong about the application's performance.

Regarding algorithms, we discovered that we were calculating some values using a “brute-force” way, not being performance-friendly. Specifically, we were computing the player’s distance from almost every component of the scene. We needed this to identify the player’s exact position to trigger the appropriate animations and sounds at the right moment. To achieve this, we placed a C# script on almost every component in the scene (for instance, doors, walls, and more). In that script, we computed the player’s distance from each object bearing this script. This computation took place for every frame. As it is evident, under no circumstances would this very costly procedure operate on an average Android device.

We decided to study and experiment with Unity3D a little more to discover colliders' meaning and usage. Colliders are 3D objects, invisible to the player (thus no rendering overhead), capable of “comprehending” when another collider interacts with them. Therefore, we removed the costly C# scripts and deployed colliders instead. That way, there was no need for computing the distance from the player during each frame. We attached a collider to the player and to every object that interested us. Then we designed some methods to execute when a collision between the player and a specific object took place (different methods for different object categories). The performance improved considerably since these methods were called only during specific collisions.

3.13.2 Cameras performance

We discovered that cameras inside the scene were still active, even when we did not need them. For instance, there is a camera in our application portal responsible for showing the users the museum world interior while being in the real world. That way, if users are outside of the museum world and look through the portal, they will view the museum world, with the portal's contents responding based on the users’ movement, thanks to that camera. A grave mistake that we made was that even when users went through the portal, the camera remained enabled, still rendering the museum's internal. That way, we had three cameras rendering sim-

ultaneously (the portal one, the player one, and the mini-map one). That was very costly in terms of performance. The same thing happened with the mini-map camera when users were outside the museum. There is no need for the museum mini-map when being outside the museum world. In our case, though, the mini-map camera was still active. By changing our code to deactivate those cameras when not required, the performance rose significantly.

3.14 The application stage so far (Virtual Reality with Oculus Go port)

After we addressed all the previously mentioned issues, we managed to port the application to the Android platform manually with success. We downloaded the ARCore SDK, switched the cameras, applied the correct settings, and built it. This procedure was not as fast as it appears, since there were many elements that we needed to consider during the port for the application to operate correctly. It took much time to explore the internet for all the actions needed to perform this port. The same applies to the Virtual Reality version with Oculus SDK since we successfully ported the application to Oculus Go.

Overall, currently, we have a version operating on Augmented Reality both in iOS and Android devices and Virtual Reality (Oculus Go). Since users do not have access to the real world in Virtual Reality, the portal's use would not make any sense. For that reason, this version starts from inside of the museum world, and there is no option to exit it. Besides, another Augmented Reality feature that would not make sense and hence it is not available is the mini-map. We thought that if there were canvases in the users' field of view, it would distract them from the museum world, thus breaking the immersion. The same applies to all the Augmented Reality version canvases, like the scoring ones and the application buttons.

Finally, we needed to reduce the complexity of the scene since Oculus Go does not have a lot of processing power. We implemented an algorithm, which hides the scene objects that are not in the same virtual room as the user. That way, the only room of the museum being populated every time is the one in which the user resides. Scene complexity is another issue, noted as future work for our framework.

3.15 Case study of the Thessaloniki PPXI application across XR

We created the museum application regarding the Industrial Museum and Cultural Centre of Thessaloniki in collaboration with the Ephorate of Modern Monuments of Thessaloniki [47]. We present the meeting phases and the comments that the people from the Ephorate supplied us with below.

3.16 Phase 1: Acquiring the museum material and development initialization

We made the initial agreement for this action as a part of the ViMM project [58]. We stayed in contact with the leading representative of the Ephorate of Modern Monuments of Thessaloniki and Ministry of Culture and Sports, Michael Tsioumas. He was the chair of the working group 4.2 focusing on the presence and new technologies, visualization, and interaction, one of the working groups of Thematic Area 4, which focused on studying methodologies and techniques of how to present information of a Virtual Museum to the visitors, depending on the target audience.

This agreement's main objective was to create an application, serving as a virtual museum for the specific museum of Thessaloniki. This application's primary purpose was to fully represent the current museum so people who will use it will feel like they are there, exploring the real museum. The requested type of application was one that would operate on mobile devices, utilizing Augmented Reality.

We received the primary 3D model of the Industrial Museum and Cultural Center of Thessaloniki in Sketchup form (.skp). We examined it thoroughly with Google's SketchUp viewer to note all available locations and generate the main plan of this application. For that, we tried answering some questions, like, for instance, "What would the application's main concept be?", "What would the main role of the user be?", "Will the application be a plain storytelling experience, or would it contain gamification elements?".

We considered various ideas and possible outcomes for this application, and finally, concluded the following:

- Since we would utilize Augmented Reality, users must be able to feel the transition from the real world to museum one. Thus, a portal to serve as the museum's entrance would be necessary.
- The virtual museum world is considerably large. Thus, its exploration might bring to the surface some restrictions for the users, as the available space in the real world does not match the virtual one. The real world may contain different boundaries, which would prevent users from exploring the virtual world. It is why a mini-map with a teleportation feature would be needed so users would always know their location in the virtual world and would be able to teleport to specific locations in the virtual world by tapping on the map.

- This application should provide users with the ability to learn available information regarding the museum as they would if they visited the real one. Thus, a storytelling feature would be necessary.
- Under no circumstances should the application be tiring or boring for users. It is why gamification elements are essential. We would add a quiz room where they will answer questions concerning the museum, evaluate themselves, and even win a virtual prize.
- The realism of the virtual museum should increase as much as possible. Thus, work is needed to calculate and improve the lighting of the museum model (global illumination) and also improve and add the correct materials to each component of the model.

Finally, a second version of the application was proposed—a pure Augmented Reality version for using inside the real museum. For the first stage, only the museum's information points were available to be placed next to some exhibits of the real museum. When users get close to each exhibit, the information point unfolds, revealing information regarding that specific exhibit to the users.

3.17 Phase 2: Meeting with museum personnel and presenting the first application draft

The first time that we presented the application [47] to the Industrial Museum and Cultural Center of Thessaloniki personnel was in September 2018, almost six months after the beginning of its development. Michael Tsioumas organized a meeting with them.

The museum personnel tried the application. They all seemed amazed. They liked the central concept of the application, the graphics seemed really nice to them, and they approved the way we presented the information to users (storytelling elements). In addition, the pure Augmented Reality version of the application was presented, which also left favorable impressions.

At the end of the presentation, a discussion followed with the museum personnel. They proposed adding 3D models of museum workers to make the virtual museum feeling more “alive”. Also, they suggested adding some animations for the machines and the Cupola furnace. Finally, Michael Tsioumas proposed that the application should be available for Android devices as well. He pointed out the importance of that ability since iOS devices are not as affordable as Android devices. By achieving this, more people will gain access to it, since the majority of them nowadays have Android devices due to their low cost.

3.18 Phase 3: Application development based on the received feedback

After receiving this useful feedback, we began evolving our application even more. In the beginning, we focused on the global illumination part by using the appropriate settings of Unity3D to better light the environment and make it look more realistic. Then, we searched for an appropriate 3D model to represent a museum worker. It was not an easy task, since although it was relatively quick to find a human 3D model, we could not find the appropriate materials (3D clothes) that a worker of that century would wear. In the end, we decided that we would create them ourselves. We managed this with the help of a 3D painting and texturing program called Substance, which helped us create a worker's clothes first draft for the museum personnel to evaluate.

The most challenging task of this phase was to make this application available for Android devices. It is the part where our SDK manager would be an outstanding contribution if we had it in that situation. Unfortunately, we had no choice but to do this part manually. The positive outcome was that we studied and noted down all the crucial options, settings, and requirements needed (for the case of ARCore) so that our project would operate correctly and without errors on an Android device.

Besides, this platform switch also helped us realize a significant drawback of our application. All this time, we were developing it for a specific iOS device, with compelling characteristics regarding its hardware. It is why the FPS (Frames-Per-Second) of our application were very high, and it was operating in high quality, providing outstanding user experience. Since Android devices greatly vary in cost, their hardware abilities also diversify. As a result, our application was not playable in some low-cost devices due to freezing, low fps, and other unpleasant effects. When we noticed all these situations, we went back to development and checked our application thoroughly. We found many performance-related bugs, which we could not have found earlier due to our previous test device's high-end capabilities. After fixing all those bugs, our application was operating on almost all Android devices (almost stands for those who support ARCore – running Android 7.0 or later version).

3.19 Phase 4: Second meeting and application presentation

After many months of work, we decided that our application's state was in excellent condition for us to present, once again, to the museum personnel. On December 24th, we decided to travel to Thessaloniki to present both of the Android versions of the application, the Augmented Reality one and the Virtual Reality one.

This time, instead of going to the museum, we went to the main offices of the Ephorate of Modern Museum of Western Macedonia to present the application. There, Michael Tsioumas

was anticipating us. We presented the Augmented Reality version initially to him and all his colleagues, operating on an android mobile device. Everyone tried it and was very satisfied with the progress. Then, we showed them the Virtual Reality version, which was operating on Oculus Go. The results were outstanding since it seems that not only they embraced the Virtual Reality version, but judging from their reaction, they seemed to like it even more, compared to the Augmented Reality one. Michael Tsioumas was so satisfied with the results that he decided to show the application to the ephorate chief. The chief tried both versions of the application, and she approved them. Like her colleagues, she seemed to like the Virtual Reality version way more. She was delighted with our cooperation results and stated that she would like to see even more results in the future.

Overall, the second meeting was a great accomplishment. Everyone was happy in the end and found out that Virtual Reality was quite preferable. Therefore, we agreed to elaborate more on the Virtual Reality version, but keep the Augmented Reality one updated. Then Michael Tsioumas gave us some suggestions to consider them for the next version of the application.

3.20 All-in-one Unity XR transition manager

Transitioning across XR, or even setting up a new project for the first time may be time-consuming for developers [57]. For instance, assume that a developer desires to start producing an Augmented Reality application for Android mobile devices. After starting Unity3D, they will have to ponder some crucial elements:

1. Is there a specific Software Development Kit (SDK) for this project that could be useful?
2. Where can this SDK be downloaded?
3. Is it open-source/free?
4. Which version is the most appropriate? (Mostly, the latest versions are the best choice, but there are also some other situations like testing or experimenting, where an older SDK version is required.)

These are the main questions that cross the developer's mind. When the developer finishes thinking about the above, what they need to do is probably open a web page and begin searching one-by-one the above questions to attain the required answers. Sometimes this could be a relatively fast procedure (especially if the developer has much experience). However, in the case of new developers, this could be quite a time-consuming procedure [57]. It is where we can prove our work to be beneficial both to experienced and inexperienced developers. We describe the first part of it below.

3.21 XR transition download manager

Another question that arises is, “What if the SDK searching and downloading procedure took place from inside Unity?”. It is why we created an SDK download manager, which is accessible through the game engine.

3.21.1 The basic architecture of XR Transition manager

The “XR Transition Manager” has a simple architecture. Its primary function is to connect to the main hosting website of the desired SDK. If the manager detects the requested version, it downloads the selected SDK through that connection and then follows the XR transition procedure. The manager handles the camera component and sets it up correctly. Developers need to specify the game-object of the camera, in any case. Overall, the manager connects with the following four components:

1. Websites hosting the SDKs.
2. Downloaded SDKs.
3. Unity3D scene.
4. Main camera component of the scene.

We can view the central architecture of the manager in Fig. 13 below.

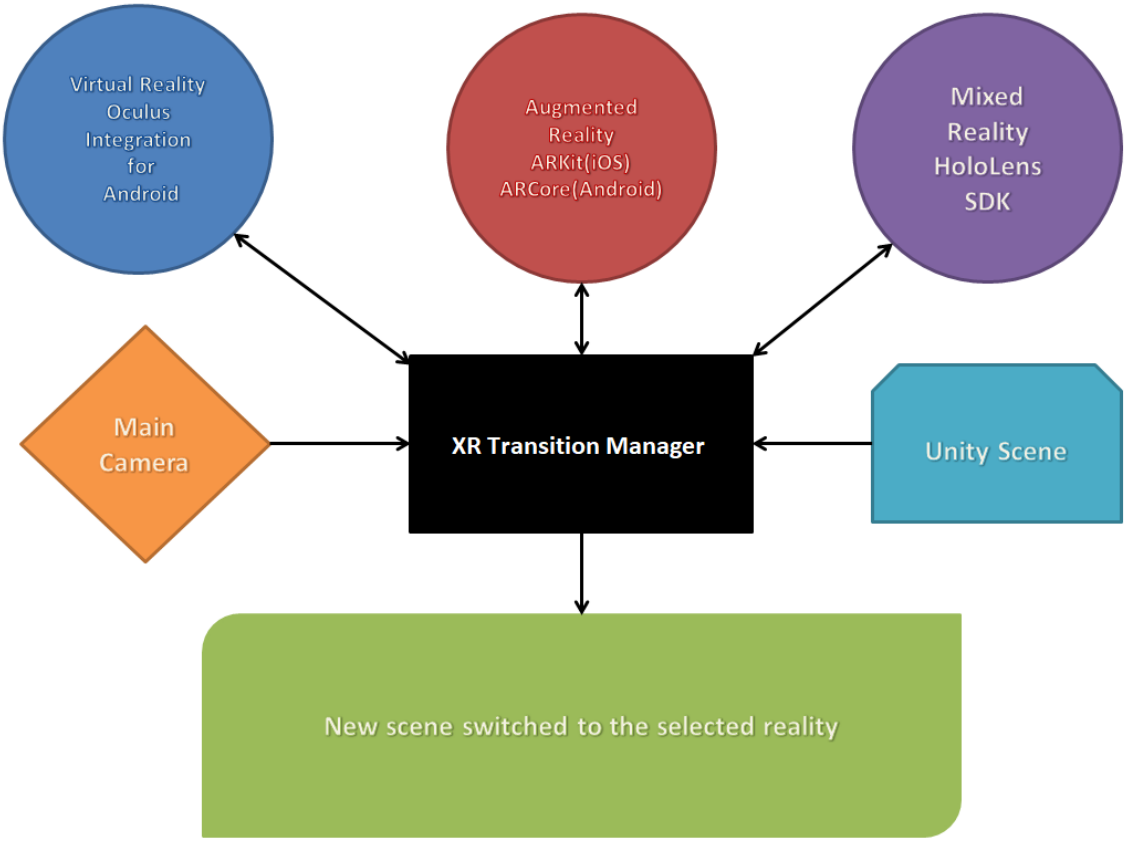

Figure 13. The basic architecture of the XR Transition manager.

3.21.2 The main code structure and logic of XR Transition Manager

"XR Transition Manager" consists of a combination of editor scripts. We designed them using the C# programming language. We divided them into three groups, namely the "Menu Script", "SDK Download & Setup Scripts" and "XR Transition Scripts". We present these groups in detail below. Due to the code being lengthy, we placed some examples in the Appendix, located at the end of this document.

3.21.2.1 Menu Script

This collection consists of one script only, namely RealitiesMenu. This script is the backbone of the whole framework. It is responsible for calling and invoking the function that developers select when they click an option from the respective Unity3D menu. For instance, if developers navigate to the Realities menu and click on the "ARCore" option under Switch Reality/Augmented Reality, the base script is responsible for calling the function, which contains all the necessary steps to perform an XR transition. See Appendix 1 for more information regarding the Menu Script code.

3.21.2.2 SDK Download & Setup Scripts

In this collection, the SDK downloading and installing scripts exist. There are four scripts in this category, one for each of the currently supported SDKs. There is one for ARKit (DownloadARKitWindow), one for ARCore (DownloadARCoreWindow), one for Mixed Reality (DownloadMixedRealityWindow), and another one for Oculus Integration (DownloadOculusWindow). All these scripts have the word "Window" appended to their name, considering the very first thing they do is to enable a window, in which developers will be able to select their desired version to install. In addition, they can check whether they would like the downloaded SDK to be installed immediately after downloading finishes from this window.

These scripts accommodate an OnGUI function, which contains everything drawn to the window from which developers select the SDK version that interests them. As its name informs us, it contains everything that exists On the Graphical User Interface.

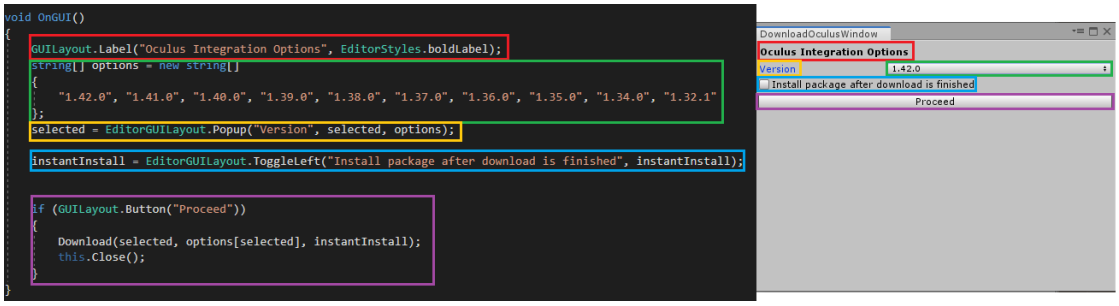

Figure 14. The contents of the OnGUI function with their respective matches to the window that the developer sees. (Each color on the left matches with its respective on the right.)

The OnGUI function follows the Download function, where the main downloading procedure takes place. Within it, there are some hardcoded links, each representing a version of the preferred SDK. Depending on the developers' SDK version decision, the script selects the appropriate link. The downloading procedure begins once the developers' internet connection is verified; otherwise, the script prints a representative error message. During downloading, we calculate and display the progress using a progress bar window. We also give developers an option to cancel the download procedure by clicking the "Cancel" button located at the window's underside.

When downloading terminates, we save the downloaded data inside the project assets folder. The "Download" function proceeds by checking whether developers chose to install the SDK instantly or not. If they chose to do so, it imports the package, using the default package importing method of Unity3D. However, the same does not hold for ARKit, where the SDK comes in compressed zip files rather than unitypackage files. For ARKit, we take extra steps to decompress the file and install it, since Unity3D does not offer a built-in system to decompress zip files automatically. We decided to use a third-party decompression tool, termed UniZip

[60], which handles the specific procedure. Once the installation step finishes, the definitions section takes place.

We take particular precautions for scripts containing code and expressions targeting specific SDKs because they will not compile unless these SDKs exist in the project. For instance, scripts that belong to the "XR Transition Scripts" collection include specific commands and utilize prefabs existing in the SDKs they represent. If these SDKs are not present during the compilation phase, it will result in compilation errors, prohibiting the whole project from operating. Thus, we created some specific definitions (one for each SDK) for the framework to identify the currently installed SDK each time. Of course, this applies only to SDKs that users installed through our framework. Using these definitions, the code that will be compiled each time will be the one that represents the currently installed SDK, thus solving the compilation problem for SDK specific code. See Appendix 2 for a detailed example of the code used for the SDK downloading and install procedure.

```
RemoveDefineIfNecessary("ARCORE_SDK", BuildTargetGroup.Standalone);
RemoveDefineIfNecessary("ARCORE_SDK", BuildTargetGroup.iOS);
RemoveDefineIfNecessary("OCULUS_SDK", BuildTargetGroup.Standalone);
RemoveDefineIfNecessary("OCULUS_SDK", BuildTargetGroup.Android);

AddDefineIfNecessary("ARKIT_SDK", BuildTargetGroup.Standalone);
AddDefineIfNecessary("ARKIT_SDK", BuildTargetGroup.iOS);
```

Figure 15. An example of managing definitions for the case of ARKit.

3.21.2.3 XR Transition Scripts

"XR Transition Scripts" is the final collection of the "XR Transition Manager". In this collection, we added four scripts, each of them being responsible for transitioning to one of the four supported platforms. Once again, the OnGUI function exists in each, setting up the window, where the developers select the main camera. Once developers press the proceed button, the script applies all the required options in the player settings menu. Then it performs a platform switch to the desired platform, constructs, and instantiates all necessary prefabs to generate a scene, which will operate successfully for the desired platform. Of course, we take some extra precautions to ensure that the manager will continue operating even if something does not go as intended. For instance, it does not take for granted that the correct SDK is installed even in this case. On the contrary, it tries to make sure that everything regarding the SDK is installed and exist. Otherwise, it interrupts the procedure by notifying the developers through an error message. See Appendix 3 for a detailed example of the code used for the XR transition procedure.

3.21.3 Downloading and installing SDKs through the XR Transition Manager

Our manager consists of editor scripts, written in C# that form a new menu, in the default Unity3D menu bar named “Realities”. A dropdown appears by pressing this button (Fig. 16), and one of the options contained inside is “Software Development Kits”. Another menu appears on the right by clicking this option, containing all available free SDKs depending on the developers’ desired platform. We currently support ARCore for Android mobile devices, ARKit for iOS devices, Oculus integration for Android VR (Oculus Go/Quest), or Windows PC. Finally, Mixed Reality toolkit for Windows Mixed Reality headsets as well as Microsoft HoloLens.

By clicking the required SDK, a window appears, granting developers a choice between different versions of this SDK (Fig. 17). Developers are offered to pick from a dropdown menu the preferred version. Also, they are prompted to click on a checkbox, in case they require the SDK to be imported/installed right after it finishes downloading. When they set all the preferences, they must click the “Proceed” button to continue.

An internet connection is vital in order to download the selected SDK. The download speed depends on the developers' internet line capabilities. Though they will experience the same speed, they would if they downloaded it from the original site.

3.22 Importance of the XR Transition SDKs download manager

As we mentioned previously, the download manager could be handy for developers while starting a new project. It speeds up the procedure and makes it exceptionally effortless. Also, it increases productivity, since it reduces the context switch between the current workspace and browsing [64] that developers may result to in case they do not remember all the necessary settings for the platform of their choice. Without our manager, developers, apart from searching and downloading the desired SDK on their own, would also have to work further to import it (unzipping the SDK and placing it in the right place). Our download manager handles this likewise. Additionally, it can be useful when developers would like to transit their application across XR. It is able to download the new SDK and get the developers up and running to develop their application in no time.

3.23 XR Transition Feature

Our “XR Transition manager” includes another fundamental and critical functionality. It downloads whichever supported AR/VR/MR SDK needed, saving a substantial amount of developers’ time. However, even though an SDK is downloaded and installed, further work is required from the developers’ side for it to work correctly for the final application to be built

and operate flawlessly. For instance, to build one that uses Google's ARCore, the following options must be set:

- The target platform must be Android.
- The default Graphics API must be either OPENGLES2 or OPENGLES3.
- The packages "AR Foundation", "Multiplayer HLAPI" and "XR Legacy Input Helper" must be installed in the current project.
- The minimum Android version must be 7.0 (API Level 24).
- The Unity3D setting "ARCore supported" must be enabled.
- Search in the SDK folders to find the specific camera default prefab and place it in the scene.

Developers must memorize all the above to set the project configuration correctly and build a sample application. It can be challenging since all these steps contain specific details (such as package names and settings) that are hard to retain. Especially for new developers, this can be rather frustrating. Usually, developers open a web browser and begin searching to find the previously mentioned settings. If they are fortunate enough, they find them in a few seconds, but there is a possibility that the correct website will not appear to them immediately. It is why our manager includes an "XR Transition" feature.

Once developers have downloaded and installed an SDK through our manager, they can decide to perform XR transition through the Realities menu. Specifically, by pressing the Realities button and then the Switch Reality one, they can select the reality of their desire (Fig. 17). When they define it, another window appears, prompting them to define their main camera game-object present in their scene (Fig. 18). In a 3D application, a camera object is mandatory for the player/user to view the scene and navigate. Thus, developers have to locate their main camera object in the Unity3D scene and place it in the appropriate box located in the window mentioned before. Once they do so, they have to click the "Proceed" button, and the SDK manager takes care of everything else. Specifically, it switches to the appropriate target platform, installs the required packages, sets up the project correctly, and instantiates the appropriate camera prefab in the scene, in the same position and rotation as the previous camera object. The latter is deactivated since it is "replaced" by the new one, but it remains in the scene, in case developers would like to do something else with it, or save it if they would like to return to the previous reality. This procedure exists in our SDK manager for switching to Augmented Reality (Android and iOS), Virtual Reality (Mobile VR – Oculus Go/Quest), and Mixed Reality (Microsoft HoloLens).

3.24 Using the XR Transition Manager

"XR Transition Manager" is a framework for Unity3D that aims to speed up the developing process of 3D applications. In section 3.11, we described our framework's primary technology. In the next sections, we will present some basic examples regarding its usage.

3.25 Downloading and installing an SDK

Once developers import all the required manager scripts correctly in their project, a menu "Realities" should appear in the menu bar of Unity3D. This menu contains four options. The first one is "Software Development Kits" and provides another four should the developers hover the mouse over it. These options are "Download ARCore", "Download ARKit", "Download Oculus Integration" and "Download Mixed Reality Toolkit".

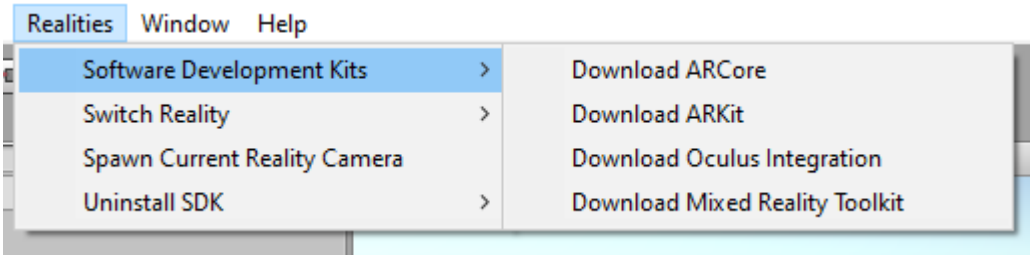

Figure 16. The supported SDKs, as they are presented in the "XR Transition Manager" menu.

Currently, our manager supports these four SDKs. By selecting one of these, developers can download the corresponding SDK. In this section, we will highlight the case of ARCore. However, the same procedure applies to the rest of the SDKs. When we click the "Download ARCore" option, the following window appears.

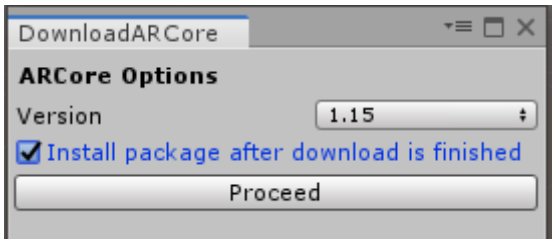

Figure 17. Choosing the version of ARCore and its installation before downloading it.

This window provides developers with the following options:

- **Version:** Developers have the option to select the SDK version of their choice. The versions are currently hardcoded in the manager. Therefore, to add newer versions, a new version of our manager has to be released as well each time.
- **Install package:** If developers check this button, the SDK is imported and installed automatically after downloading. Otherwise, either a .unitypackage or a .zip file will appear in the project path for the developer to install/import it manually.

When developers set the version and the package install option, the procedure continues by clicking the “Proceed” button.

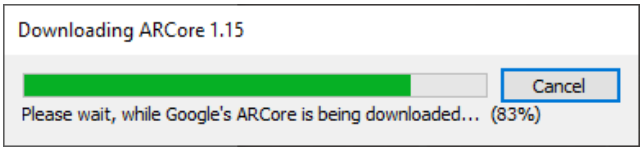

Figure 18. The progress of SDK downloading.

The window depicted above indicates the progress of the download procedure. Developers also have the option to stop it at any time by clicking the cancel button. In case of success, the following window appears, indicating that the download procedure was successful.

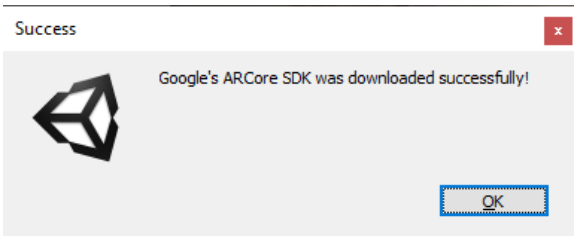

Figure 19. Success message for the successful download of SDK.

After developers click the OK button, depending on their initial choice, the SDK will be imported and installed, or they will receive a .unitypackage file containing the SDK, to install it by themselves.

3.26 Switching reality

The second option that the “Realities” menu provides is transitioning the project across XR depending on the developers’ choice. Again, in this example, we will focus on ARCore, but the same procedure applies across XR. As visible from the image below, there are three options under the “Switch Reality” tab.

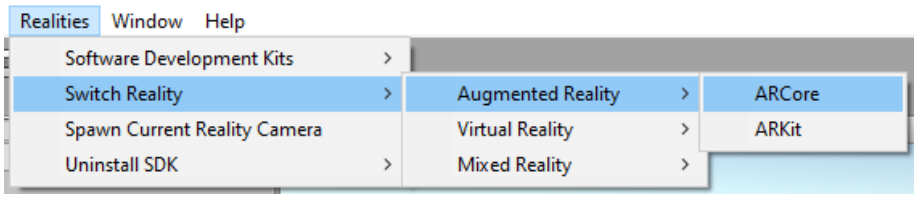

Figure 20. The supported realities of switching to, depending on the selected SDK/platform.

- Augmented Reality: This reality is mostly for mobile devices equipped with a camera. It is because, as mentioned previously, Augmented Reality offers partial immersion. For Augmented Reality, the two sub-options that are currently supported by our manager are ARCore and ARKit.

- Virtual Reality: This reality is for an application operating on a Head-Mounted-Display (HMD). Currently, our manager supports only Oculus Integration, rendering the supported devices to be mobile VR and Oculus Go/Quest.
- Mixed Reality: This reality is for VR/AR applications operating on Microsoft devices. Currently, we tested this option on Microsoft HoloLens (Holographic Augmented Reality).

Developers should consider switching to a scene supporting the SDK that they have downloaded. If they have not downloaded the correct SDK, an error message will notify them about it, and the manager will prompt them to download it.

When developers select the reality they would like their project to switch into, the following window appears.

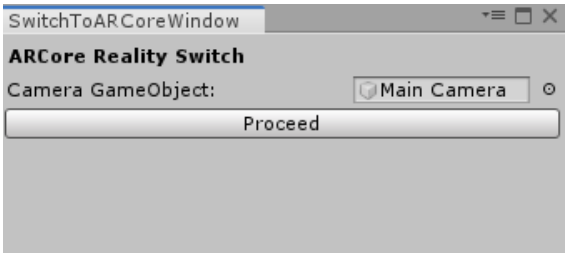

Figure 21. Settings window of switching to Augmented Reality for ARCore/Android.

The next step is to specify the main camera of their application, which serves as the users' eyes. Once it is set and developers click the “Proceed” button, the procedure of XR transitioning begins. The time developers have to wait afterward depends on the size of their project. This action performs a platform switch, sets all the required settings for the current SDK, and adds the main prefabs/objects in the scene required by the SDK for a successful and working build. Of course, after the operation finishes, developers can add/delete SDK related prefabs according to their desire. The XR transitioning procedure performs all the necessary options in a few seconds; otherwise, the developers would have to do it manually.

3.27 Uninstalling an SDK

The uninstalling feature could not have been missing from our SDK manager. There is a separate option in the “Realities” menu regarding uninstalling an already downloaded SDK. The image below depicts it.

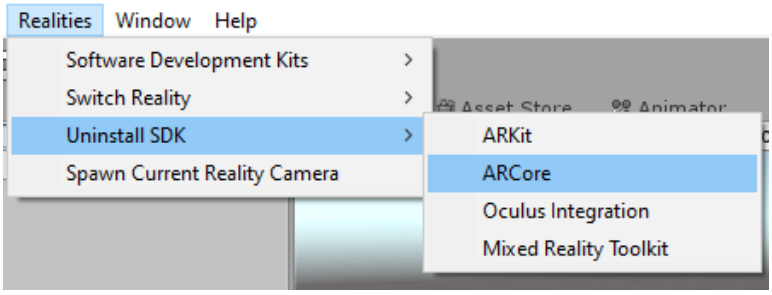

Figure 22. The SDK uninstall feature.

Someone could submit the following question. "Why should a developer do that through the 'Realities' menu and not by just selecting the SDK folder and manually deleting it?". That would be the same as deleting an unwanted program's folder in our computers. The program would not be there, but a lot of information and data would remain internally. The same holds for our SDK manager. Developers may delete the SDK folder separately, but all the options and settings our manager performed during this SDK installation would remain. These additional settings reset by our manager are the main reason for developers to have a "healthy" and "ready to operate" project after the SDK removal. See Appendix 4 for a detailed example of the code used for uninstalling an SDK.

3.28 Automatic reality working camera

Sometimes, developers would like to place a working camera on their current project and reality, without needing to undergo an XR transitioning procedure (testing purposes, for instance). Our manager handles this case since it offers an option to spawn an XR specific camera on demand. We call this option "Spawn Current Reality Camera". When selected, our manager automatically detects the platform and installed SDK developers are working on and adds the corresponding test-ready camera to the scene. See Appendix 5 for more information regarding the code used for this feature.

3.29 Development of XR Transition Manager

The development procedure of the "XR Transition Manager" was relatively smooth, with a few challenges that took place along the way.

3.29.1 The SDK handling of XR Transition Manager

The "XR Transition Manager's" central idea is to host several valid download links to each version of various SDKs and provide functionalities of installing and setting them up. It aims to ease the developers to quickly develop and build their applications without any SDK related errors. For each SDK, we include links to its stable versions in the code used for downloading it, in case the developers select the specific version. We use the UnityWebRequest API to vali-

date the connection and proceed to the actual download. When it finishes, the downloaded file is recorded temporarily in the assets folder to be imported and installed to the project. We import the file using the ImportPackage utility of Unity3D. Unity3D offers the ability to make a prompt appear to developers when importing a package, enumerating the package contents to specify which components to install. We decided to disable this prompt since we install only the SDK's mandatory components (thus, everything is needed) to save time during installation and importing. After these procedures end, we delete the temporary downloaded file, leaving only the installed components.

3.29.2 Challenges Faced During Development

The main challenges that we faced during the development of the “XR Transition Manager” were two:

Firstly, Apple’s ARKit SDK is distributed in .zip form, instead of .unitypackage. However, the ImportPackage utility of Unity3D would not be successful in installing such files. To overcome this issue, we had to install Apple’s ARKit without using the utilities of Unity3D. We investigated different approaches by writing our C# functions but to no avail. Eventually, we found a third-party Unity3D plugin called UnityZip (UniZip); we imported it to the project and used this to decompress .zip files.

Secondly, another challenge that we faced was downloading the SDK packages. In the beginning, we considered downloading all the versions of each SDK, save them locally (as part of our manager), and install them, reducing the download time to zero. In the end, this approach did not produce any favorable results, as some SDKs were quite large (the size of Oculus SDK consists of some hundreds of MBs and has many versions), thus forming a space bottleneck. Moreover, this would not have been practical, as it would not offer portability. So, we tried experimenting with different ways to download these SDKs through C# code. We then discovered that using UnityWebRequest was our best bet since it was easy to utilize and practical. However, when we used it, we could not find the downloaded file. It was another challenge for us since we believed that this utility saved the file after downloading it. After some experimenting and searching, we found out that the file needed an extra step to save it successfully. We simply had to check the download operation and find out when it finishes. Then we had to retrieve the downloaded data from the web request structure and save it to our desired path using the File utility.

4 Results and Conclusions

In this section, we will summarize our work. It was a fascinating topic, which we enjoyed working on. Since there is no actual limit to what can be done or added to this work to make it even more helpful and meaningful to the developers, we will discuss some potential ideas to be added in the future.

4.1 Summary

In this work, we initially presented a framework for manually transitioning a digital heritage application across XR (from VR to holographic AR). We presented how we transferred a Virtual Reality application to holographic Augmented Reality and the challenges we faced during this platform switch. Afterward, we introduced the main problem with application transitioning across XR in a 3D application and discussed why this should occur automatically. Moreover, we showcased our latest work on cultural heritage, an application for the Industrial Museum and Cultural Center of Thessaloniki [47]. Finally, we described the "XR Transition Manager" in detail, its functions, and how it works.

4.2 Evaluation

To examine our system's overall user experience, we conducted a preliminary user-based evaluation with ten users (eight of them were proficient Unity3D developers, whereas the other two were relatively new to Unity3D). The main research questions were the following:

- How time-consuming is the manual SDK installation/reality switch/SDK uninstallation for a project in Unity3D?
- Is the SDK download function of our system preferable in comparison to the manual downloading procedure?
- Is the "Switch Reality" feature of our system preferable compared to the manual platform switch and settings application procedure?
- Is the SDK Uninstallation feature of our system preferable in comparison to the manual deletion of each SDK?
- Is the "Spawn Current Reality Camera" function of our manager useful?
- Would a developer prefer to use our manager for their Unity3D projects?

4.2.1 Methodology and participants

We divided the experiment into four different parts, each for one of the questions previously described. It is worth noting that all the participants were software developers, and they were familiar with Unity3D.

4.2.1.1 Part 1: Manual Installation of ARCore SDK

In this part, we asked the participants to manually download and install an SDK (specifically ARCore SDK). They had to do it manually at first. They had to search for tutorials (in case they did not know how to install it). To succeed in this part, they had to include all the necessary files and directories of the SDK needed to compile successfully (thus, no compile errors in the Unity3D editor).

4.2.1.2 Part 2: XR Transition Testing

In this part, we asked the participants to switch to the reality supported by the SDK they downloaded in the previous step. Again, they had to do it manually. To succeed in this step, they had to press the "Play" button of the Unity3D editor to run without any errors. For this part, we measured both the time they needed to initiate a platform switch manually and apply all the necessary settings. Finally, we received their feedback on the intricacy of the procedure.

4.2.1.3 Part 3: SDK Uninstallation

In this part, we asked the participants to uninstall the SDK they installed in the previous steps. To succeed in this step, they had to reverse the Unity3D editor to its initial state (when they started the evaluation). They had to do it manually. In the end, we measured the time they needed and received their feedback regarding the complexity of the procedure.

4.2.1.4 Part 4: Performing the tasks with "XR Transition Manager"

As this evaluation's final step, we asked the participants to perform the previous actions again, but this time, using our manager. This procedure was speedy (ten to twenty seconds) for users to find and click the appropriate buttons on the "Realities" menu. We should note that the SDK download time and system execution time for platform switch were not calculated, since that depends on the system and internet connection speed. In the end, we asked them to provide us with their feedback about the current procedure, and finally, we asked them if they would prefer to use our "XR Transition Manager" in their future Unity3D projects.

4.2.1.5 Questionnaire

On the next page, we present the questionnaire that we used.

Realities SDK Manager Evaluation

This is a form for the evaluation of the Realities SDK Manager tool for Unity3D created for the purposes of my Msc Thesis.

1. How much time did you need to manually install Google's ARCore to your Unity3D project?

_____

2. How easy you believe it was to install Google's ARCore through "Realities SDK Manager"?

	1	2	3	4	5	6	7	8	9	10	
Extremely Difficult	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Extremely Easy

3. How much time did you need to manually switch to Augmented Reality with ARCore and sucessfully run the project?

_____

4. How easy you believe it was to switch reality through "Realities SDK Manager"?

	1	2	3	4	5	6	7	8	9	10	
Extremely Difficult	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Extremely Easy

5. How much time did you need to manually uninstall (remove from your project) Google's ARCore?

_____

6. How easy you believe it was to uninstall an SDK through "Realities SDK Manager"?

	1	2	3	4	5	6	7	8	9	10	
Extremely Difficult	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Extremely Easy

7. How much useful did you find the "Spawn Current Reality Camera" function?

	1	2	3	4	5	6	7	8	9	10	
Not at all	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Extremely Useful

8. How likely is that you would you use "Realities SDK Manager" in your Unity3D projects?

	1	2	3	4	5	6	7	8	9	10	
Not at all	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Extremely likely

4.2.2 Results

4.2.2.1 Part 1: Manual Installation of ARCore SDK

For this part, we measured the average time they needed to install ARCore SDK to their Unity3D project. We told them to search the internet freely for tutorials and steps on how to do this. For this part, time was our variable. It is important to note that we did not measure the time needed for downloading the SDK for this step, or the SDK system installation procedure. We present the results for this part in the table below.

Participant	Hours	Minutes	Seconds	Time Decrease Percentage (for an average time of 15 s)
#1	0	20	0	98.75%
#2	0	8	10	96.94%
#3	0	9	55	97.48%
#4	0	5	8	95.13%
#5	0	9	40	97.41%
#6	0	8	33	97.08%
#7	0	7	22	96.61%
#8	0	5	47	95.68%
#9	0	17	9	98.54%
#10	0	7	36	96.71%

Table 2. Participants' results for the first part of our evaluation.

4.2.2.2 Part 2: XR Transition Testing

For this part, we also measured the average time the participants needed to manually perform a transition across XR and prepare a sample scene of ARCore to run in Unity3D editor. Again, the participants were free to search for tutorials online on how to do this, in case they did not know. We did not measure the time needed for platform switch, since it depends on the system hardware. We present the results in the table below.

Participant	Hours	Minutes	Seconds	Time Decrease Percentage (for an average time of 15 s)
#1	0	10	0	97.50%

#2	0	5	0	95.00%
#3	0	1	10	78.57%
#4	0	2	16	88.97%
#5	0	4	43	94.70%
#6	0	12	31	98.00%
#7	0	18	27	98.64%
#8	0	8	23	97.02%
#9	0	22	34	98.89%
#10	0	14	32	98.28%

Table 3. Participants' results for the second part of our evaluation.

4.2.2.3 Part 3: SDK Uninstallation

It is the final part where we measured time. In this part, we wrote down the time the participants needed to remove ARCore SDK from their Unity3D project and return the project to its initial state. Again, system-dependent time was not measured (platform switch). We present the results in the following table.

Participant	**Hours**	**Minutes**	**Seconds**	**Time Decrease Percentage (for an average time of 15 s)**
#1	0	5	21	95.33%
#2	0	2	7	88.19%
#3	0	0	40	62.50%
#4	0	4	6	93.90%
#5	0	7	11	96.52%
#6	0	1	58	87.29%
#7	0	2	38	90.51%
#8	0	1	35	84.21%
#9	0	4	31	94.46%
#10	0	2	14	88.81%

Table 4. Participants' results for the third part of our evaluation.

4.2.2.4 Part 4: Performing the tasks with “XR Transition Manager”.

After participants finished doing the tasks manually, we requested to perform them again using the “XR Transition Manager”. It was a relatively quick procedure because they execute these actions with a button press. So that would be a few seconds (ten to twenty) for each participant. Finally, we asked the participants to give us their feedback by answering some questions. We present their feedback below.

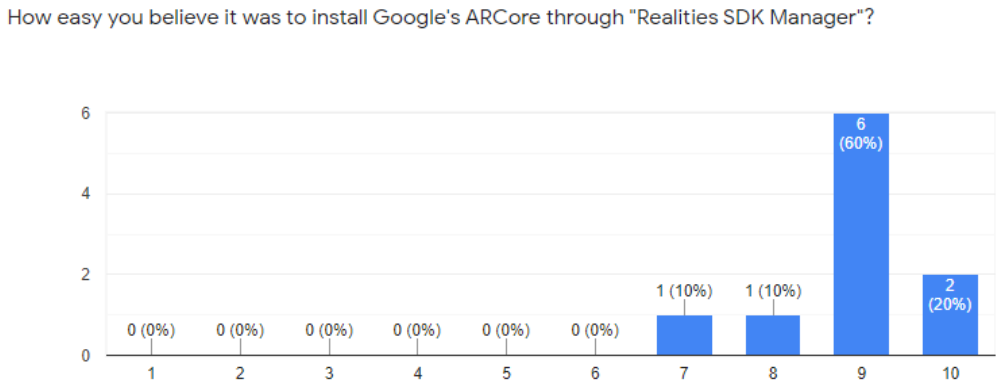

Figure 23. Results regarding the installation of ARCore through our manager.

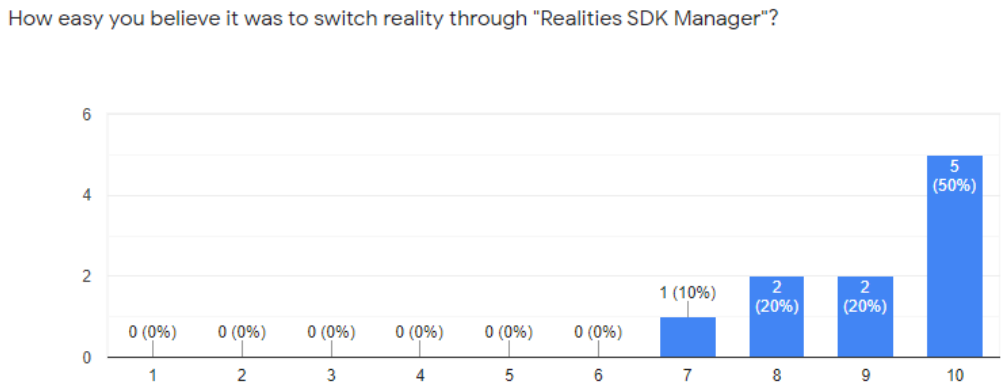

Figure 24. Results regarding the transition across the XR procedure through our manager.

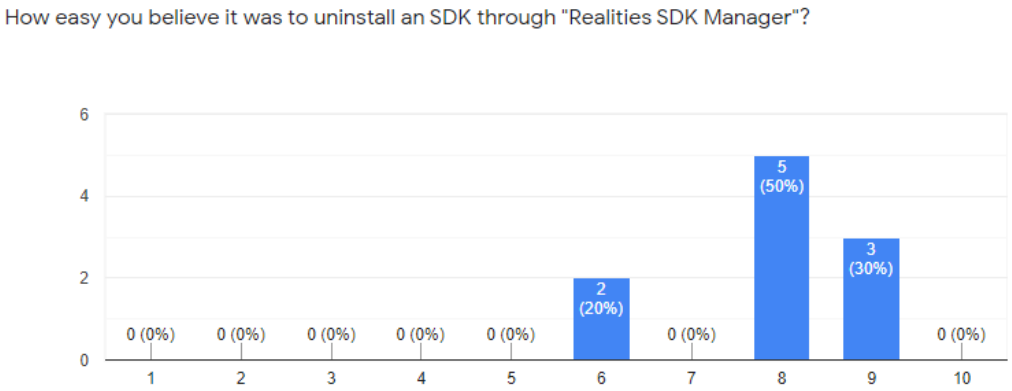

Figure 25. Results regarding the SDK uninstallation through our manager.

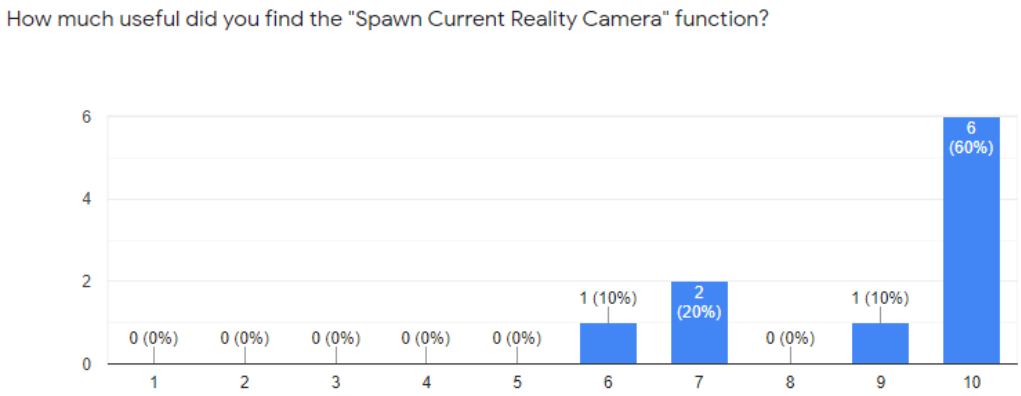

Figure 26. Results regarding the "Spawn Current Reality Camera" function of our manager.

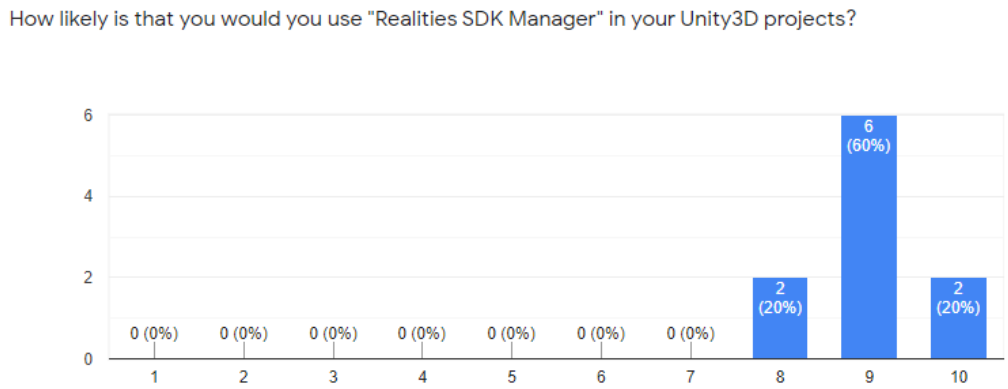

Figure 27. Participants' opinion on the overall effectiveness of our manager.

4.2.3 Discussion

The current evaluation brought us fascinating results. Some participants were quick enough for both installation and transitioning across XR as well, whereas some participants needed much time (about twenty minutes). For a programmer's development, twenty minutes is quite a long time for such a procedure, which our manager seeks to avoid. The same applies to the XR transition procedure as well (twenty-two minutes was the longest). The uninstallation procedure was quicker, although some cases needed some time (five or seven minutes, for instance, is considered quite long for such a case).

Then, participants tried our manager. The amount of time needed for all these tasks was between ten to twenty seconds per task for everyone. By taking the average of this time (15 seconds), we are able to calculate a percentage of saved time for each participant.

We calculated the time decrease percentage (in seconds) for each participant and each evaluation step. We calculated each percentage using the following formula:

$$\% = \frac{ManualTime - AverageTransitionFrameworkTime}{ManualTime} * 100$$

“Manual Time” stands for the time (in seconds) that participants needed to do a requested action without using our framework. “Average Transition Framework Time” stands for the average time needed for participants to perform an action using our framework. It is equal to 15 seconds, the average of a ten to twenty seconds window that each participant needed. These results are vital because our framework managed to save between 62.50% (worst case) and 98.89% of the participants’ time (best case). Conclusively, our framework saved more than 62.50% of the participants’ time (above average), which is a win for us. Our current aim is to raise the worst-case percentage even more in order to benefit even more developers.

Apart from the time-consuming scenario, we also wished to know our participants’ thoughts about our manager and, more importantly, its ease of utilization. By observing Fig.23 and Fig. 24, we understand that all participants gave a score of seven-plus out of ten for the SDK installing and XR transition tasks. For the uninstallation procedure (Fig. 25), participants graded our manager with eight and nine out of ten. Two participants graded it with a six. We believe that is because our manager's SDK uninstallation procedure does not handle the cases where a plugin (.dll) might be used from Unity3D when users try to uninstall the SDK. This prevents our manager from entirely removing the SDK. It is a challenge for us to solve in the future.

The majority of participants delivered a ten to our “Spawn Current Reality Camera function” (Fig 26). Two participants gave it a seven out of ten, and one participant gave it a six. Belatedly, the participants seemed to approve our manager in general, by stating that they would use it in their Unity3D projects. They all gave a score of eight-plus out of ten when they were asked this question (Fig. 27).

4.2.4 Evaluation Conclusion

To conclude the evaluation, the participants appeared to approve our framework, which made us more eager to upgrade it in the future continually. Our manager received more than a 60% score for each participant's feedback, which is quite a positive result. Besides, our manager saved more than 62.50% of the time they spent. It is why we intend to work eagerly for the next months to stabilize it further and add more abilities that will be beneficial for developers. We present these additions in the following section.

5 Future Work

We wanted to add some additions/features to our framework but could not due to the lack of time. We believe that these features would make the “XR Transition Manager” even more helpful to developers. We divided these features into the following sections.

5.1.1 Saving Time

Saving time is one of the most critical elements of our manager, if not the most prominent. It already saves a reasonable amount of time by managing four different SDKs and transcending across XR based on them.

Until now, our manager successfully substitutes the project camera (which is selected by the developer). It adds all the necessary objects in the scene required by the current SDK to work correctly. It also applies all the necessary settings and installs the packages needed. However, we do nothing regarding the controls of the application. For instance, suppose we would like to transform an application operating on Oculus Go (mobile virtual reality) to run on a mobile device with Augmented Reality. Oculus Go has a controller, with which users can perform various interactions in their applications. Suppose an application moves to Augmented Reality (from Virtual Reality) by utilizing our manager. In that case, users would not be able to perform any interactions since such mobile devices use completely different ways of interaction. For instance, most mobile devices use the touch screen to receive input from users. We want to provide our manager with a way to map all the supported functions from the input module of the originating reality to the target reality. For example, the pressing of a button in the Oculus Go controller could be mapped to a simple touch on a mobile device's touchscreen or a pinch gesture for HoloLens.

5.1.2 Expanding availability

We want to benefit as more developers as possible with this framework. At its current state, the number of developers who can profit is somehow limited. For developers to use this framework, they must be familiar with Unity3D, and their target device for their application must be a mobile device, Oculus Go, or Microsoft HoloLens. We want to put an end to this boundary by expanding the number of game engines on which our manager will be installed and the number of supported devices and SDKs. As a result, our foremost future objective is to make our manager available for the Unreal game engine since many developers use it. Moreover, we would like to provide support to all desktop Virtual Reality devices, SDKs, and mobile headsets/devices. That would significantly increase the number of developers who can be ben-

efited by our manager. Finally, we would like to adjust the manager properly, like converting it to a single package or DLL file, to distribute way more efficiently.

5.1.3 Complete reality transformation

Our framework may currently transit a 3D application across XR, but it does not consider any theoretical elements that the target reality may have. We want to add this feature in the future. This feature will scan the application scene's contents and perform some necessary changes regarding the target reality. For instance, if we move an Augmented Reality application containing a portal to Virtual Reality, that portal would be of no use. That is because Virtual Reality provides users with full immersion, and thus they do not have access to the real world. In that case, our framework should be able to detect such portals and disable them.

Another example would be switching from Virtual Reality to Mixed Reality and Microsoft HoloLens [48]. In this case, if our application contains a room surrounded by virtual walls, these walls should be detected and removed. That is because, in Mixed Reality, the virtual world blends with the real. Thus, the contents of the 3D room should blend with the contents of the real room. As a result, there should be no boundaries (i.e., virtual walls surrounding the room), as it would ruin the users' immersion.

5.1.4 Smart Performance Adaptation

Our manager's main objective is to make the application porting procedure easier for developers, so that eventually, in the future, more and more 3D applications will be available for many devices. All those devices are different, though, in terms of hardware and overall performance. It is something that we would also like our manager to consider. We describe this in detail below.

5.1.4.1 Complexity of the models

The 3D scenes of such projects contain different kinds and sizes of 3D models. Sometimes, 3D models tend to be very complicated because they might contain many vertices or complicated geometry. The more complex a model is, the less likely the application will operate smoothly on devices with lower specifications. That is a challenge that we would like to relieve the developers from. We plan to upgrade our manager to consider the device that the application will operate on and apply the necessary quality settings for the application to run smoothly on that device. We will also examine the case of applying reduction algorithms to those meshes, to diminish the geometry complexity and make them lighter for low specs devices.

5.1.4.2 Lighting

One common solution to overcome low performance, concerning lighting, especially in low specs devices, is static lighting. It makes the scene less realistic but increases performance. We want to add some global illumination calculation algorithms to our manager. Again, depending on the target device, if the device is a low specs one, our manager would provide light and realistic real-time illumination algorithms. Even in those devices, real-time lighting will be available. We believe that such a feature will significantly improve the application quality while operating on low specs devices and render them highly realistic.

References

[1] Christoforidou S., Kondylidou A., Mesochoriti E., Syrgianni A., Valavanidou A.: The Islahane of Thessaloniki, a contemporary monument, a witness to technical education as a means of vocational training of orphan children.

[2] Ioannides, M., Magnenat-Thalmann, N., Papagiannakis, G., (Eds).: Mixed Reality and Gamification for Cultural Heritage, Springer, DOI: 10.1007/978-3-319-49607-8, (2017).

[3] Hamari, J., Koivisto, J., Sarsa, H.: Does Gamification Work? -- A Literature Review of Empirical Studies on Gamification, Proc. of the 47th Hawaii Int'l Conference on System Sciences (HICSS), IEEE Computer Society, Waikoloa, HI, USA, 3025-3034, (2014).

[4] Devindra. "Oculus Quest review: VR freedom comes at a cost". Engadget. Retrieved 11-12-2019.

[5] ARKit Vs. ARCore – How they compare against each other? https://www.itfirms.co/arkit-vs-arcore-how-they-compare-against-each-other/. Accessed 11 Aug 2018

[6] Davies, Chris (May 1, 2015). "HoloLens hands-on: Building for Windows Holographic". http://www.slashgear.com/hololens-hands-on-building-for-windows-holographic-01381717/. Accessed 23 Feb 2020.

[7] "Microsoft Hololens hardware". http://www.microsoft.com/microsoft-hololens/hardware . Accessed 23 Feb 2020.

[8] Alex Kipman, Seth Juarez (April 30, 2015). Developing for HoloLens. http://channel9.msdn.com/Events/Build/2015/C9-08. Event occurs at 00:07:15. Accessed 23 Feb 2020.

[9] Hachman, Mark (May 1, 2015). "Developing with HoloLens: Decent hardware chases Microsoft's lofty augmented reality ideal". http://www.pcworld.com/article/2917613/developing-with-hololens-decent-hardware-chases-microsofts-lofty-augmented-reality-ideal.html. Accessed 23 Feb 2020.

[10] Hollister, Sean (January 21, 2015). "Microsoft HoloLens Hands-On: Incredible, Amazing, Prototype-y as Hell". https://gizmodo.com/project-hololens-hands-on-incredible-amazing-prototy-1680934585. Accessed 23 Feb 2020.

[11] Microsoft HoloLens: The Science Within - Spatial Sound with Holograms. https://www.youtube.com/watch?v=aB3TDjYklmo. February 29, 2016. Accessed 23 Feb 2020.

[12] Holmdahl, Todd (April 30, 2015). "BUILD 2015: A closer look at the Microsoft HoloLens hardware". https://blogs.windows.com/devices/2015/04/30/build-2015-a-closer-look-at-the-microsoft-hololens-hardware/. Accessed 23 Feb 2020.

[13] Liarokapis, F., Sylaiou, S., Basu, A., Mourkoussis, N., White, M., Lister, P.F. An Interactive Visualisation Interface for Virtual Museums, Proc. of the 5th International Symposium on Virtual Reality, Archaeology and Cultural Heritage, Eurographics Association, Brussels, Belgium, 6-10 Dec, 47-56, 2004.

[14] Sandor, C., Fuchs, M., Cassinelli, A., Li, H., Newcombe, R., Yamamoto, G., Feiner, S. (2015). Breaking the Barriers to True Augmented Reality.

[15] Jung, Timothy & Tom Dieck, M. Claudia & Lee, Hyunae & Chung, Namho. (2016). Effects of Virtual Reality and Augmented Reality on Visitor Experiences in Museum. 10.1007/978-3-319-28231-2_45.

[16] Papaeftymiou M., Kanakis E.M., Geronikolakis E., Nochos A., Zikas P., Papagiannakis G., "Rapid Reconstruction and Simulation of Real Characters in Mixed Reality Environments", Digital Cultural Heritage Lecture Notes in Computer Science, Vol. 10605, 267-276, 2018.

[17] Papagiannakis, G., Geronikolakis, E., Pateraki, M., Bendicho, V.M., Tsioumas, M. Sylaiou, S., Liarokapis, F., Grammatikopoulou, A., Dimitropoulos, K., Grammalidis, N., Partarakis, N., Margetis, G., Drossis, G., Vassiliadi, M., Chalmers, A., Stephanidis, C., Thalmann, N. (2018). Mixed Reality Gamified Presence and Storytelling for Virtual Museums.

[18] Ioannides, Marinos, et al., eds. Digital Heritage. Progress in Cultural Heritage: Documentation, Preservation, and Protection: 7th International Conference, EuroMed 2018, Nicosia, Cyprus, October 29–November 3, 2018, Proceedings. Vol. 11196. Springer, 2018.

[19] Zikas, P., Bachlitzanakis, V., Papaefthymiou, M., Kateros, S., Georgiou, S., Lydatakis, N., Papagiannakis G., Mixed reality serious games for smart education. In European Conference on Games Based Learning 2016. ECGBL'16, 2016.

[20] Abate, A., Barra, S., Galeotafiore, G., Díaz, C., Aura, E., Sánchez, M., Mas, X., Vendrell Vidal, E. (2018). An Augmented Reality Mobile App for Museums: Virtual Restoration of a Plate of Glass: 7th International Conference, EuroMed 2018, Nicosia, Cyprus, October 29–November 3, 2018, Proceedings, Part I. 10.1007/978-3-030-01762-0_47.

[21] Anderson, E.F., McLoughlin, L., Liarokapis, F., Peters, C., Petridis, P., de Freitas, S.: Serious Games in Cultural Heritage. In: Ashley, M., Liarokapis, F. (eds.) The 10th International Symposium on Virtual Reality, Archaeology and Cultural Heritage VAST – State of the Art Reports (2009).

[22] Papaefthymiou, M., Papagiannakis, G., "Gamified Augmented and Virtual reality character rendering and animation enabling technologies". In: Ioannides, M., Magnenat-Thalmann, N., Papagiannakis, G. (Eds.) Mixed Reality and Gamification for Cultural Heritage, pp. 333-357. Springer, DOI: 10.1007/978-3-319-49607-8, (2017).

[23] Pedersen, I., Gale, N., Mirza-Babaei, P., Reid, S.: More than Meets the Eye: The Benefits of Augmented Reality and Holographic Displays for Digital Cultural Heritage. Journal on Computing and Cultural Heritage (JOCCH), 10(2), 11, (2017).

[24] Drossis, G., Ntelidakis, A., Grammenos, D., Zabulis, X., Stephanidis, C.: Immersing users in landscapes using large scale displays in public spaces. In: International Conference on Distributed, Ambient, and Pervasive Interactions (pp. 152-162). Springer, Cham, (2015, August).

[25] Tisserand, Y., Magnenat-Thalmann, N., Unzueta, L., Linaza, M.T., Ahmadi, A., O'Connor, N.E., Zioulis, N., Zarpalas, D., Daras, P.: Preservation and Gamification of Traditional Sports. In: Mixed Reality and Gamification for Cultural Heritage, Ioannides, M., Magnenat-Thalmann, N., Papagiannakis, G. (Eds.), (pp. 421-446), Springer International Publishing, (2017).

[26] http://apelab.ch/spatialstories . Accessed 6 January 2018

[27] Stefanidi, E., Arampatzis, D., Leonidis, A., Papagiannakis, G. (2019). BricklAyeR: A Plat-form for Building Rules for AmI Environments in AR. 10.1007/978-3-030-22514-8_39.

[28] Pfeiffer-Leßmann, N., Pfeiffer, T. ExProtoVAR: A Lightweight Tool for Experience-Focused Prototyping of Augmented Reality Applications Using Virtual Reality. (2018) 10.1007/978-3-319-92279-9_42.

[29] Pau Xiberta, Imma Boada, A new e-learning platform for radiology education (RadEd), Computer Methods and Programs in Biomedicine, Volume 126, 2016, Pages 63-75, ISSN 0169-2607, 10.1016/j.cmpb.2015.12.022.

[30] Kotis K. ARTIST - a reAl-time low-effoRt mulTi-entity Interaction System for creaTing re-usable and optimized MR experiences (2019). Research Ideas and Outcomes 5: e36464. https://doi.org/10.3897/rio.5.e36464.

[31] Bugalia, N., Kumar, S., Kalra, P., Choudhary, S.: Mixed reality based interaction system for digital heritage. In: Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry-Volume 1 (pp. 31-37). ACM, (2016, December).

[32] Dong, Y., Webb, M., Harvey, C., Debattista, K., Chalmers, A.: Multisensory Virtual Experience of Tanning in Medieval Coventry. In: EUROGRAPHICS Workshop on Graphics and Cultural Heritage. 27-29, Graz, Austria, (September 2017).

[33] Gimeno, J., Olanda, R., Martinez, B., Sanchez, F. M.: Multiuser augmented reality system for indoor exhibitions. In: IFIP Conference on Human-Computer Interaction (pp. 576-579). Springer, Berlin, Heidelberg, (2011, September).

[34] Grammenos, D., Michel, D., Zabulis, X., Argyros, A. A.: PaperView: augmenting physical surfaces with location-aware digital information. In: Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction (pp. 57-60). ACM, (2011, January).

[35] Javornik, A., Rogers, Y., Gander, D., Moutinho, A.: MagicFace: Stepping into Character through an Augmented Reality Mirror. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (pp. 4838-4849). ACM, (2017, May).

[36] Kitsikidis, A., Kitsikidis, A., Dimitropoulos, K., Uğurca, D., Bayçay, C., Yilmaz, E., Tsalakanidou, F., ... & Grammalidis, N.: A game-like application for dance learning using a natural human computer interface. In: International Conference on Universal Access in Human-Computer Interaction (pp. 472-482). Springer, Cham, (2015, August).

[37] Kosmalla, F., Zenner, A., Speicher, M., Daiber, F., Herbig, N., Krüger, A.: Exploring Rock Climbing in Mixed Reality Environments. In: Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (pp. 1787-1793). ACM, (2017, May).

[38] Koutsabasis, P., Vosinakis, S.: Kinesthetic interactions in museums: conveying cultural heritage by making use of ancient tools and (re-) constructing artworks, Virtual Reality - Special Issue: VR and AR Serious Games, Springer, (2017).

[39] Liarokapis, F., Kouřil, P., Agrafiotis, P., Demesticha, S., Chmelík, J., Skarlatos, D.: 3D Modelling and Mapping For Virtual Exploration of Underwater Archaeology Assets, Proc. of the 7th Int'l Workshop on 3D Virtual Reconstruction and Visualization of Complex Architectures and Scenarios, ISPRS, Napflio, Greece, 425-431, (2017).

[40] Margetis, G., Ntoa, S., Antona, M., Stephanidis, C.: Interacting with augmented paper maps: a user experience study 12th biannual Conference of the Italian SIGCHI Chapter (CHITALY 2017). 18th -20th September, Cagliari, Italy, (2017).

[41] Nakevska, M., van der Sanden, A., Funk, M., Hu, J., Rauterberg, M.: Interactive storytelling in a mixed reality environment: the effects of interactivity on user experiences. Entertainment Computing, (2017).

[42] Papagiannakis, G., Schertenleib, S., Ponder, M., Arevalo-Poizat, M., Magnenat-Thalmann, N., Thalmann, D., "Real-Time Virtual Humans in AR Sites", IEE Visual Media Production (CVMP04), pp. 273-276, London, (March 2004).

[43] Zikas, P., Bachlitzanakis, V., Papaefthymiou, M., Kateros, S., Georgiou, S., Lydatakis, N., Papagiannakis, G., "Mixed Reality Serious Games for smart education", European Conference on Games Based Learning 2016, ECGBL'16, Paisley, Scotland, October 2016.

[44] Liarokapis F., Petridis P., Andrews D., de Freitas S. (2017) Multimodal Serious Games Technologies for Cultural Heritage. In: Ioannides M., Magnenat-Thalmann N., Papagiannakis G. (eds) Mixed Reality and Gamification for Cultural Heritage. Springer, Cham.

[45] Bertuzzi, Juan, and Khaldoun Zreik. "Mixed Reality Games - Augmented Cultural Heritage." In Augmented Culture: Proceedings of the 15th Iberoamerican Congress of Digital Graphics, 304-307. SIGraDi. Santa Fe, Argentina, 2011.

[46] Ott, Michela & Dagnino, Francesca & Pozzi, Francesca & Yilmaz, Erdal & Tsalakanidou, Filareti & Dimitropoulos, Kosmas & Nikos, Grammalidis. (2015). Serious Games to Support Learning of Rare 'Intangible' Cultural Expressions.

[47] Efstratios Geronikolakis, Michael Tsioumas, Stephanie Bertrand, Athanasios Loupas, Paul Zikas, George Papagiannakis: New Cross/Augmented Reality Experiences for the Virtual Museums of the Future. EuroMed (1) 2018: 518-527.

[48] Efstratios Geronikolakis, Paul Zikas, Steve Kateros, Nick Lydatakis, Stelios Georgiou, Mike Kentros, George Papagiannakis: A True AR Authoring Tool for Interactive Virtual Museums. CoRR abs/1909.09429 (2019).

[49] Stephanie Bertrand, Martha Vassiliadi, Paul Zikas, Efstratios Geronikolakis, George Papagiannakis: From Readership to Usership and Education, Entertainment, Consumption to Valuation: Embodiment and Aesthetic Experience in Literature-based MR Presence. CoRR abs/1910.10019 (2019).

[50] Paul Zikas, Nick Lydatakis, Steve Kateros, George Papagiannakis, "Scenior: An Immersive Visual Scripting system of Gamified Training based on VR Software Design Patterns", 2019, arXiv:1909.05719v1.

[51] George Papagiannakis, Paul Zikas, Nick Lydatakis, Steve Kateros, Mike Kentros, Efstratios Geronikolakis, Manos N. Kamarianakis, Ioanna Kartsonaki, Giannis Evangelou: MAGES 3.0: Tying the knot of medical VR. CoRR abs/2005.01180 (2020).

[52] Oculus Rift, Wikipedia https://en.wikipedia.org/wiki/Oculus_Rift (last accessed July 2, 2020).

[53] ARCore – Quickstart for Android, https://developers.google.com/ar/develop/unity/quickstart-android (last accessed July 26, 2020).

[54] Unity MARS, https://unity.com/products/unity-mars, (last accessed July 26, 2020).

[55] Quiver - 3D Coloring App, https://apps.apple.com/us/app/quiver-3d-coloring-app/id650645305, (last accessed August 31, 2020).

[56] Papagiannakis, G., Lydatakis, N., Kateros, S., Georgiou, S., and Zikas, P., 2018. Transforming Medical Education and Training with VR using M.A.G.E.S. In Proceedings of Siggraph Asia '18 Posters, Tokyo, Japan, December 04-07, 2018 https://doi.org/10.1145/3283289.3283291

[57] How to port your app to another platform, https://mwdn.com/port-app-another-platform/, (last accessed September 2, 2020)

[58] ViMM EU Project that received funding from the European Union's Horizon 2020 Programme as Coordination and Support Action, under GA n° 727107, https://www.vi-mm.eu/, (last accessed September 2, 2020)

[59] Oculus Go review, https://www.techradar.com/reviews/oculus-go, (last accessed September 2, 2020)

[60] UnityZip, https://github.com/tsubaki/UnityZip, (last accessed September 2, 2020)

[61] ARCore Supported Devices, https://developers.google.com/ar/discover/supported-devices, (last accessed September 3, 2020)

[62] The XRSI Definitions of Extended Reality (XR), XR Data Classification Framework, XR-DCF Public Working Group XR Safety Initiative, California, USA.

[63] Introducing ARKit 4, https://developer.apple.com/augmented-reality/arkit/, (last accessed September 10, 2020)

[64] Software Development: Productivity and Context switching, https://medium.com/@mayuminishimoto/software-development-productivity-and-context-switching-66f99b388033, (last accessed September 14, 2020)

[65] Percentage Decrease: Formula & Calculation, https://study.com/academy/lesson/percent-decrease-formula-calculation.html, (last accessed September 14, 2020)

Appendix 1 – The XR Transition Manager Menu

This menu is the backbone of the whole transition framework. That is because it is the initial communication bridge between the framework and the developers. It calls the appropriate function based on the developers' choices.

```
public class RealitiesMenu : EditorWindow
{
    [MenuItem("Realities/Software Development Kits/Download ARCore")]
    static void DownloadARCore()
    [MenuItem("Realities/Software Development Kits/Download ARKit")]
    static void DownloadARKit()
    [MenuItem("Realities/Software Development Kits/Download Oculus Integration")]
    static void DownloadOculusIntegration()
    [MenuItem("Realities/Software Development Kits/Download Mixed Reality Toolkit")]
    static void DownloadHoloToolKit()
    [MenuItem("Realities/Switch Reality/Augmented Reality/ARCore")]
    static void SwitchToARCore()
    [MenuItem("Realities/Switch Reality/Augmented Reality/ARKit")]
    static void SwitchToARKit()
    [MenuItem("Realities/Switch Reality/Virtual Reality/Mobile VR")]
    static void SwitchToMobileVR()
    [MenuItem("Realities/Switch Reality/Mixed Reality/HoloLens")]
    static void SwitchToHoloLens()
    [MenuItem("Realities/Spawn Current Reality Camera")]
    static void SpawnCamera()
    [MenuItem("Realities/Uninstall SDK/ARKit")]
    static void DeleteARKit()
    [MenuItem("Realities/Uninstall SDK/ARCore")]
    static void DeleteARCore()
    [MenuItem("Realities/Uninstall SDK/Oculus Integration")]
    static void DeleteOculus()
    [MenuItem("Realities/Uninstall SDK/Mixed Reality Toolkit")]
    static void DeleteMRToolkit()
}
```

Each "MenuItem" keyword in this code represents an option offered from the "Realities" menu when clicked. The dashes represent all the sub-menus. According to which option is clicked by the developers, the appropriate function is called. For instance, if developers click on the "Download ARKit" option, under "Software Development Kits", the "DownloadARKit" function will be called, initiating the download procedure for ARKit.

Appendix 2 – The code responsible for downloading an SDK (Case of Oculus SDK)

In this appendix, we present the code used for an application transition to Mobile VR, utilizing the Oculus SDK.

The following code downloads and installs (depending on developer choice) the Oculus Integration SDK.

```
public class DownloadOculusWindow : EditorWindow
{
    static float downloadDataProgress = 0.0f;
    int selected = 0;
    bool instantInstall = false;
    bool canceled = false;

    void OnGUI()
    {
        GUILayout.Label("Oculus Integration Options", EditorStyles.boldLabel);
        string[] options = new string[]
        {
            "1.42.0", "1.41.0", "1.40.0", "1.39.0", "1.38.0", "1.37.0", "1.36.0", "1.35.0", "1.34.0",
"1.32.1"
        };
        selected = EditorGUILayout.Popup("Version", selected, options);

        instantInstall = EditorGUILayout.ToggleLeft("Install package after download is finished", instan-
tInstall);

        if (GUILayout.Button("Proceed"))
        {
            Download(selected, options[selected], instantInstall);
            this.Close();
        }
    }

    void Download(int selected, string version, bool install)
    {
        string url = "";
        switch (selected)
        {
            case 0:
                url =
"https://securecdn.oculus.com/binaries/download/?id=2792787640732487&access_token=OC%7C1196467420370658%7C";
                break;
            case 1:
                url =
"https://securecdn.oculus.com/binaries/download/?id=2725569690787616&access_token=OC%7C1196467420370658%7C";
                break;
            case 2:
                url =
"https://securecdn.oculus.com/binaries/download/?id=2644800202197899&access_token=OC%7C1196467420370658%7C";
                break;
            case 3:
                url =
"https://securecdn.oculus.com/binaries/download/?id=2581006521910601&access_token=OC%7C1196467420370658%7C";
                break;
            case 4:
                url =
"https://securecdn.oculus.com/binaries/download/?id=2574886002522653&access_token=OC%7C1196467420370658%7C";
                break;
            case 5:
                url =
"https://securecdn.oculus.com/binaries/download/?id=2574880252523228&access_token=OC%7C1196467420370658%7C";
                break;
            case 6:
                url =
"https://securecdn.oculus.com/binaries/download/?id=2574875629190357&access_token=OC%7C1196467420370658%7C";
                break;
            case 7:
                url =
"https://securecdn.oculus.com/binaries/download/?id=2574871702524083&access_token=OC%7C1196467420370658%7C";
                break;
            case 8:
```

```
                url =
"https://securecdn.oculus.com/binaries/download/?id=2574863995858187&access_token=OC%7C1196467420370658%7C";
                break;
            case 9:
                url =
"https://securecdn.oculus.com/binaries/download/?id=2574859885858598&access_token=OC%7C1196467420370658%7C";
                break;
            default:
                break;

        }
```

```
        using (UnityWebRequest webRequest = UnityWebRequest.Get(url))
        {
            var operation = webRequest.SendWebRequest();

            if (webRequest.isNetworkError)
            {
                Debug.LogError("There was the following error during downloading Oculus Integration version
" + version + ": " + webRequest.error + ".\nPlease consider choosing a different version.");
            }

            while (!operation.isDone)
            {

                downloadDataProgress = webRequest.downloadProgress * 100;
                if(EditorUtility.DisplayCancelableProgressBar("Downloading Oculus Integration " + version,
"Please wait, while Oculus Integration is being downloaded... (" + (int)downloadDataProgress + "%)", down-
loadDataProgress / 100.0f))
                {
                    canceled = true;
                    webRequest.Abort();
                    break;
                }
            }
            EditorUtility.ClearProgressBar();
```

```
            if (canceled)
            {
                EditorUtility.DisplayDialog("Cancelled", "The SDK downloading process was interrupted by the
user.", "OK");
                this.Close();
                return;
            }
            else if (operation.webRequest.error != null)
            {
                EditorUtility.DisplayDialog("Error", "There was an error while downloading ARCore SDK.
Please try again or choose another version.", "OK");
                canceled = true;
            }
```

```
            if(!canceled)
                File.WriteAllBytes(Application.dataPath + "\\OculusIntegration_" + version +
".unitypackage", webRequest.downloadHandler.data);

        }
        if (!canceled && instantInstall)
        {
            AssetDatabase.ImportPackage(Application.dataPath + "\\OculusIntegration_" + version +
".unitypackage", false);
            FileUtil.DeleteFileOrDirectory(Application.dataPath + "\\OculusIntegration_" + version +
".unitypackage");
            EditorUtility.DisplayDialog("Success", "The Oculus SDK was downloaded successfully!", "OK");


            RemoveDefineIfNecessary("ARKIT_SDK", BuildTargetGroup.Standalone);
            RemoveDefineIfNecessary("ARKIT_SDK", BuildTargetGroup.iOS);
            RemoveDefineIfNecessary("ARCORE_SDK", BuildTargetGroup.Standalone);
            RemoveDefineIfNecessary("ARCORE_SDK", BuildTargetGroup.Android);

            AddDefineIfNecessary("OCULUS_SDK", BuildTargetGroup.Standalone);
            AddDefineIfNecessary("OCULUS_SDK", BuildTargetGroup.Android);
        }
    }

    public static void AddDefineIfNecessary(string _define, BuildTargetGroup _buildTargetGroup)
    {
        var defines = PlayerSettings.GetScriptingDefineSymbolsForGroup(_buildTargetGroup);

        if (defines == null) { defines = _define; }
        else if (defines.Length == 0) { defines = _define; }
        else { if (defines.IndexOf(_define, 0) < 0) { defines += ";" + _define; } }

        PlayerSettings.SetScriptingDefineSymbolsForGroup(_buildTargetGroup, defines);
```

```
    }

    public static void RemoveDefineIfNecessary(string _define, BuildTargetGroup _buildTargetGroup)
    {
        var defines = PlayerSettings.GetScriptingDefineSymbolsForGroup(_buildTargetGroup);

        if (defines.StartsWith(_define + ";"))
        {
            // First of multiple defines.
            defines = defines.Remove(0, _define.Length + 1);
        }
        else if (defines.StartsWith(_define))
        {
            // The only define.
            defines = defines.Remove(0, _define.Length);
        }
        else if (defines.EndsWith(";" + _define))
        {
            // Last of multiple defines.
            defines = defines.Remove(defines.Length - _define.Length - 1, _define.Length + 1);
        }
        else
        {
            // Somewhere in the middle or not defined.
            var index = defines.IndexOf(_define, 0, System.StringComparison.Ordinal);
            if (index >= 0) { defines = defines.Remove(index, _define.Length + 1); }
        }

        PlayerSettings.SetScriptingDefineSymbolsForGroup(_buildTargetGroup, defines);
    }
}
```

This code is separated into five parts, according to the color of the rectangle it is enclosed to:

- **Blue:** This code is responsible for rendering the Unity3D editor window, presenting the options for the SDK version. It creates a simple UI for developers to choose the version of the selected SDK from a dropdown menu instantly.
- **Yellow:** In this part, the download links for each version of the specific SDK are located.
- **Green:** This code represents the download procedure. It begins instantly, without having the developers navigate to external websites to do so. At the same time, the download progress is displayed through a progress-bar window.
- **Red:** Error handling, in case of download failure or cancellation by the developers.
- **Black:** This code handles the SDK installation. It removes the definitions of SDKs that are not needed while adding the definition of the recently downloaded one. It also automatically imports the downloaded package.

Also, if the reality that the developers are interested in requires some extra packages to be installed in the project, it is done by the following code (example taken from the case of ARCore, where the package multiplayer-hlapi was required):

```
Client.Add("com.unity.multiplayer-hlapi");
Client.Embed("com.unity.multiplayer-hlapi");
```

Finally, the code presented here (and most importantly that of the blue, green and black rectangles) is quite beneficial for the developers. It reduces context switching (from Unity3D editor and code to browser), which has positive results to productivity [64].

Appendix 3 – The code responsible for performing a reality transition (to mobile Virtual Reality)

The following code switches to mobile Virtual Reality using the Oculus Integration SDK.

```
public class SwitchToMobileVR : EditorWindow
{
    public GameObject source;
    Transform playerTransform;
    int selected = 0;
    bool cameraFound = false;
    UnityEngine.Rendering.GraphicsDeviceType[] graphicsAPI;
    string[] sdks = new string[1];

    void OnGUI()
    {
        GUILayout.Label("Mobile VR Switch", EditorStyles.boldLabel);
        EditorGUILayout.BeginHorizontal();
        EditorGUILayout.LabelField("Camera GameObject: ");
        source = EditorGUILayout.ObjectField(source, typeof(Object), true) as GameObject;
        EditorGUILayout.EndHorizontal();
        graphicsAPI = new UnityEngine.Rendering.GraphicsDeviceType[1];
        graphicsAPI[0] = UnityEngine.Rendering.GraphicsDeviceType.OpenGLES3;
        sdks[0] = "Oculus";
        if (GUILayout.Button("Proceed"))
        {
#if OCULUS_SDK
            PlayerSettings.SetGraphicsAPIs(BuildTarget.Android, graphicsAPI);
            PlayerSettings.Android.minSdkVersion = AndroidSdkVersions.AndroidApiLevel24;
            PlayerSettings.virtualRealitySupported = true;
            PlayerSettings.Android.ARCoreEnabled = false;
            PlayerSettings.vuforiaEnabled = false;
            PlayerSettings.SetVirtualRealitySDKs(BuildTargetGroup.Android, sdks);

            EditorUserBuildSettings.SwitchActiveBuildTarget(BuildTargetGroup.Android, BuildTarget.Android);

            if (GameObject.Find("OVRPlayerController") != null)
            {
                EditorUtility.DisplayDialog("Player Camera Exists", "The prefab for Mobile VR default camera
was found in the scene.", "OK");
                this.Close();
                return;
            }
            EditorUtility.DisplayDialog("Warning", "OVRPlayerController contains the Character Controller
component, which will lead to your camera falling unless there is a collider beneath it. Please acknowledge
that fact for your development procedure.", "Got it");

            Object prefab = AssetData-
base.LoadAssetAtPath("Assets\\Oculus\\VR\\Prefabs\\OVRPlayerController.prefab", typeof(GameObject));

            if (prefab == null)
            {
                EditorUtility.DisplayDialog("Error", "The operation could not continue, because some essen-
tial files of Oculus Integration were not found.", "OK");
                source.SetActive(true);
                this.Close();
                return;
            }

            GameObject player = Instantiate(prefab) as GameObject;

            if (source != null)
            {
                playerTransform = source.transform;
                player.transform.position = playerTransform.position;
                player.transform.rotation = playerTransform.rotation;

            }
            player.name = "OVRPlayerController";
            source.SetActive(false);
            this.Close();
#else
            if (EditorUtility.DisplayDialog("Oculus integration SDK not found", "It seems that the SDK for
Oculus integration was not found in this project. Would you like to download it now?", "Yes", "No"))
            {
                DownloadOculusWindow window = (DownloadOculusWin-
dow)EditorWindow.GetWindow(typeof(DownloadOculusWindow));
                window.Show();
```

```
            }
            else
            {
                EditorUtility.DisplayDialog("Error", "The operation could not continue, because Oculus inte-
gration SDK was not found.", "OK");
                this.Close();
            }
            return;
#endif
        }
    }
}
```

This class performs a transition to Virtual Reality for Oculus mobile devices. Developers only need to specify the camera game-object in their scene and click the "Proceed" button. Then follows the automated transition of the Unity3D project to Virtual Reality and Android platform. We separated this code into four sections according to their roles, specified by the color of the rectangle they are enclosed to.

- **Blue:** This code checks if Oculus SDK has been defined (installed) and then performs all the necessary settings that the project requires to be successfully compiled and operate on the target device.
- **Yellow:** This code is responsible for checking if the corresponding camera already exists in the scene (OVRPlayerController prefab in this case).
- **Green:** This code searches the Oculus SDK and loads (or constructs if necessary) the appropriate camera game-object and places it inside the project scene. This renders the transition procedure very easy and effortless since developers do not have to remember all these settings or search for the appropriate camera components. This is rather beneficial, especially for SDKs like ARKit, which does not offer a ready-to-utilize prefab for the main camera game-object (at the time of writing). Contrariwise, developers need to modify the default camera game-object of Unity3D by attaching specific ARKit components/scripts to render it functional.
- **Red:** This code handles the case of Oculus SDK absence, either by stopping the whole procedure or prompting the developer to download the appropriate SDK.

The same code structure holds for each reality transition. The only variable elements are the specific reality's settings, camera construction and error handling. As a result, we are able to support more SDKs and platforms in the future easily.

The code in the blue and green rectangles is essential since it prevents developers from searching online for such settings. Apart from this being a time-consuming procedure, it is also counterproductive since developers will have to keep switching their context from the Unity3D editor and code to the browser for information searching on how to do these things [64]. This code takes care of all these settings and game-object set up.

Appendix 4 – The Code for an SDK Uninstallation

The code responsible for an SDK uninstallation is shown below. This is the case for ARCore, but it is almost the same for the other SDKs.

```
static void DeleteARCore()
    {
        try
        {
            foreach (GameObject obj in UnityEngine.Object.FindObjectsOfType(typeof(GameObject)))
            {
                if (obj.name.Equals("ARCore Device") || obj.name.Equals("Plane Generator") ||
obj.name.Equals("Point Cloud") || obj.name.Equals("PlaneDiscovery"))
                {
                    DestroyImmediate(obj);
                }
            }


            RemoveDefineIfNecessary("ARCORE_SDK", BuildTargetGroup.Android);
            RemoveDefineIfNecessary("ARCORE_SDK", BuildTargetGroup.Standalone);
            EditorUserBuildSettings.SwitchActiveBuildTarget(BuildTargetGroup.Standalone, BuildTar-
get.StandaloneWindows64);
            FileUtil.DeleteFileOrDirectory(Application.dataPath + "\\GoogleARCore");
            FileUtil.DeleteFileOrDirectory(Application.dataPath + "\\PlayServicesResolver");
            Cleanup("GoogleARCore");
            Cleanup("PlayServicesResolver");
            RemoveEmptyFoldersMenuItem("GoogleARCore");
            RemoveEmptyFoldersMenuItem("PlayServicesResolver");
            EditorUtility.DisplayDialog("Uninstall Successful", "ARCore was uninstalled successfully. You
may need to refresh your project hierarchy for the SDK folders to disappear.", "OK");
        }
        catch (Exception e)
        {
            if(e is DirectoryNotFoundException)
            {
                EditorUtility.DisplayDialog("ARCore Not Found", "ARCore was not found in this project.",
"OK");
            }
            if (e is UnauthorizedAccessException)
            {
                EditorUtility.DisplayDialog("Cannot Delete", "Some assets could not be deleted. Make sure
nothing is keeping a hook on them, like a loaded DLL for example.", "OK");
            }
        }
    }
```

Initially, this code searches the project scene and destroys game-objects related with the camera for the specific SDK and target platform. Then it removes the definition of this SDK from the game engine to mark its uninstallation. Afterwards, it performs a platform switch to the default one (PC, Mac & Linux Standalone) and finally removes all SDK related files from the project directory. Developers are informed about the procedure success, or its failure, in case of an exception, due to exception handling. Exceptions could either be that the SDK was not found, or some dlls could not be deleted because the game engine is currently using them. In that case, developers will have to close the game-engine and then remove them. This is a challenge we faced and we plan on overcoming it in the future.

Appendix 5 – The code for the "Spawn Current Reality Camera" Feature

This code is responsible for spawning a camera game-object based on the reality the developer is at the time of calling. The reasons for this are many, from simple testing, to the simultaneous existence of many cameras in the scene.

```
if ARCORE_SDK
GameObject Camera = new GameObject();
        Camera.AddComponent<Camera>();
        Camera.AddComponent<TrackedPoseDriver>();
        Cam-
era.GetComponent<TrackedPoseDriver>().SetPoseSource(TrackedPoseDriver.DeviceType.GenericXRDevice,TrackedPose
Driver.TrackedPose.ColorCamera);
        Camera.GetComponent<TrackedPoseDriver>().updateType = TrackedPoseDriver.UpdateType.Update;
        Camera.GetComponent<TrackedPoseDriver>().UseRelativeTransform = true;
        Camera.AddComponent<GoogleARCore.ARCoreBackgroundRenderer>();
        Camera.GetComponent<GoogleARCore.ARCoreBackgroundRenderer>().BackgroundMaterial = AssetData-
base.LoadAssetAtPath("Assets\\GoogleARCore\\SDK\\Materials\\ARBackground.mat", typeof(Material)) as
Material;
        Camera.name = "ARCore Camera";
        EditorUtility.DisplayDialog("Google's ARCore Camera", "The system detected that your current project
is set for Google's ARCore. The appropriate camera will be spawned.", "OK");
#endif
#if ARKIT_SDK
        GameObject Camera = new GameObject();
        Camera.AddComponent<Camera>();
        Camera.AddComponent<UnityARVideo>();
        DirectoryInfo dir = new DirectoryInfo(Application.dataPath + "\\ARKit");
        DirectoryInfo[] info = dir.GetDirectories();
        UnityEngine.Object Material = AssetDatabase.LoadAssetAtPath("Assets\\ARKit\\" + info[0].Name +
"\\Assets\\UnityARKitPlugin\\Plugins\\iOS\\UnityARKit\\Materials\\YUVMaterial.mat", typeof(Material));
        Camera.GetComponent<UnityARVideo>().m_ClearMaterial = Material as Material;
        Camera.AddComponent<UnityARCameraNearFar>();
        Camera.GetComponent<Camera>().clearFlags = CameraClearFlags.Depth;
        Camera.name = "ARKit Camera";
        EditorUtility.DisplayDialog("Apple's ARKit Camera", "The system detected that your current project
is set for Apple's ARKit. The appropriate camera will be spawned.", "OK");
#endif
#if OCULUS_SDK
        UnityEngine.Object Camera = AssetData-
base.LoadAssetAtPath("Assets\\Oculus\\VR\\Prefabs\\OVRCameraRig.prefab", typeof(GameObject)) as GameObject;
        Camera = Instantiate(Camera) as GameObject;
        Camera.name = "OVRCameraRig";
        EditorUtility.DisplayDialog("Oculus Mobile Camera", "The system detected that your current project
is set for Oculus Mobile. The appropriate camera will be spawned.", "OK");#endif
#if !ARKIT_SDK && !ARCORE_SDK && !OCULUS_SDK
  GameObject Camera = new GameObject();
        Camera.AddComponent<Camera>();
        Camera.name = "Main Camera";
        EditorUtility.DisplayDialog("Default Desktop Camera", "The system detected that your current project
has not been set for a specific reality. The default camera of Unity3D will be spawned.", "OK");

#endif
```

Initially, the code checks if the project is currently set for operating with ARCore SDK, then ARKit SDK and finally, OCULUS_SDK. Depending on what the situation is, the code constructs and spawns in the scene the appropriate working camera object, loading all the necessary components from the respective SDK. In case none of the above SDKs is present, the default camera object of Unity3D will be spawned.

Appendix 6 – Installation guide for XR Transition Manager

This guide explains how to install the XR Transition Manager to a Unity3D Project. The manager is tested and working with the latest Unity3D (2019.2.11f1 at the time of writing).

To install the manager in your project, kindly follow the steps below:

1. Open the Unity3D project, in which you would like to install the SDK manager.
2. In the root directory of the project (Assets folder), create a new folder and name it "Editor", if it does not exist already (Fig. 28).
3. Extract the contents of the "Editor" folder of the package "RealitiesSDKManager.zip" to the "Editor" folder in the current project.
4. In the root directory of the project (Assets folder), create a new folder and name it "Plugins", if it does not exist already (Fig. 29).
5. Extract the contents of the "Plugins" folder of the package "RealitiesSDKManager.zip" to the "Plugins" folder in the current project.
6. Wait until the "Realities" menu appears on the menu bar of Unity3D (Fig. 30).
7. XR Transition Manager is installed successfully.

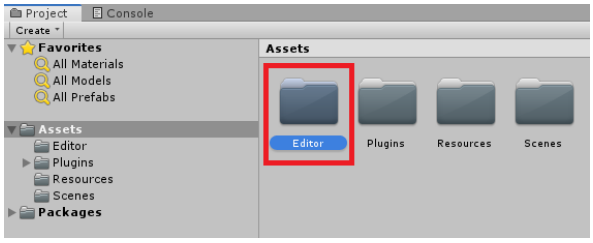

Figure 28. A folder with the name "Editor" must exist in the "Assets" directory of the project.

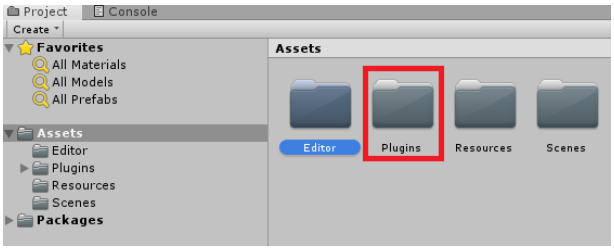

Figure 29. A folder with the name "Plugins" must exist in the "Assets" directory of the project.

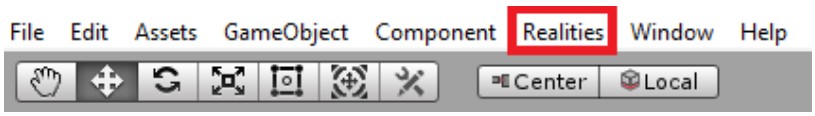

Figure 30. Indication that Realities SDK Manager is successfully installed and ready to be used.