Holo-box: Multi-Level Augmented Reality Glanceable Interfaces for Machine Shop Guidance using Eye and Hand Tracking Interactions

Grigorios Daskalogrigorakis

Masters Thesis
School of Electrical and Computer Engineering
Technical University of Crete
June 2022

Supervisor: Prof. Mania Katerina, Technical University of Crete
Committee: Assoc. Prof. McNamara Ann, Texas A&M University
Committee: Assoc. Prof. Samoladas Vasilis, Technical University of Crete

Abstract

In this work, we employ Serious Games and Augmented Reality as tools for manufacturing training inside a machine shop. First, we analyze the progress of modern manufacturing processes leading up to the 4th industrial revolution, commonly known as Industry 4.0. We tackle shortcomings in relation to the use of simulations and Serious Games as training tools as well as the use of Augmented Reality in manufacturing. Finally, we analyze Glanceable Augmented Reality and its technical challenges.

In the first part of this work, we propose a gamified approach to visualising manufacturing missions for the implementation of a Serious Game focused on G-code programming for milling and turning missions for undergraduate students as a standalone training system without the need of a supervisor. Our gamified manufacturing mission design was also translated for use through AR inside the real world machine shop.

Glanceable User Interfaces for Augmented Reality (AR) reveal virtual content "at a glance" to provide rapid information retrieval, often based on gaze interaction. They are ideal when the augmented content covers a small proportion of the view space, but when the size of virtual content grows, the potential to occlude the real-world increases provoking safety concerns.

In this work, we present *Holo-Box*, a novel interaction system for AR combining Glanceable interfaces and world-based 3D interfaces across three Levels-Of-Detail, including progressively more information and visuals. Holo-box uses a combination of eye-gaze and hand interactions, focusing on user safety. A 2D Glanceable interface facilitates rapid information retrieval at a glance, while extended 3D interfaces provide interaction with denser content and 3D objects. Holo-Box couples blink-based and gaze-based interactions to minimize errors arising from the Midas Touch Problem. While applicable across domains, the Holo-Box interface is designed and optimized for performing manufacturing tasks in the real world. We evaluated the Holo-Box interface using an object selection task of a manufacturing process. Participants completed subsequent tasks faster using Holo-Box, employing incrementally smaller LOD interfaces over time. The perceived accuracy of Holo-Box gaze-based inputs was high, even when the device's eye tracker accuracy was coarse.

Contents

Related Publications

1. [1] Kirakosian, S., Daskalogrigorakis, G., Maravelakis, E., & Mania, K. (2021, August). Near-contact Person-to-3D Character Dance Training: Comparing AR and VR for Interactive Entertainment. In 2021 IEEE Conference on Games (CoG) (pp. 1-5). IEEE.

2. [2] Daskalogrigorakis, G., Kirakosian, S., Marinakis, A., Nikolidakis, V., Pateraki, I., Antoniadis, A., & Mania, K. (2021, April). G-Code Machina: A Serious Game for G-code and CNC Machine Operation Training. In 2021 IEEE Global Engineering Education Conference (EDUCON) (pp. 1434-1442). IEEE.

3. [3] Daskalogrigorakis, G., McNamara, A., & Mania, K. (2021). Holo-Box: Level-of-Detail Glanceable Interfaces for Augmented Reality. In ACM SIGGRAPH 2021 Posters (pp. 1-2).

4. Daskalogrigorakis, G., McNamara, A., Marinakis, A. Antoniadis, A., & Mania, K. (submitted 2022) Glance-Box: Multi-LOD Glanceable Interfaces for Machine Shop Guidance using Blink and Hand Interaction. IEEE.

1 Introduction

Training new personnel for manufacturing is often a tedious operation. Practice under real-world machinery is expensive, consuming vital system's resources and, in some cases, dangerous for the trainee [4]. G-code programs guide the machining process, written in a computer or machine controller and loaded to a machine. A safer alternative for training uses simulated factories or training environments. The concept of a digital factory can be defined as an interactive 3D environment supporting modeling, communications and operation of manufacturing [5]. Previous studies indicate that virtual training is effective as the trainee is guided through complex product manufacturing [6], [7]. While there are plenty of gamified virtual simulations that aid in manufacturing training and mainly focus on safety tasks [8] or manual operation [9], the task of creating a serious game focused on writing G-code is technically challenging. Potential trainees often have no previous programming knowledge. It is significant that they learn how to write efficient G-code programs, not just correct ones.

Augmented Reality (AR) provides an enhanced vision of the world by seamlessly integrating virtual computer-generated elements with real-world environments [10]. AR applications are prominent in complex, innovative workplaces. The manufacturing floors of modern machine shops are increasingly employing AR technology as the evolution of the industry moves towards the *industry 4.0 model* [11]. A limiting factor of AR Head-Mounted Displays (HMDs) stems from the narrow Field of View (FoV) available in current hardware. This narrow FoV restricts the amount of virtual content that can be viewed at one time which, in turn, occludes real-world elements, critical to the task at hand or the user's safety.

Workers employ AR to safely retrieve virtual information at any point eliminating the distraction and spatial limitations of traditional 2D screens[12]. AR applications need compelling, intuitive interfaces that do not impede or endanger the user to be effective in this space [13]. One approach is to use *Glanceable User Interfaces* that can retrieve information "at a glance" based on gaze detection when using an AR HMD with integrated eye-tracking [14]. Glanceable interfaces prove helpful in cases when the user is focused on a real-world task that directly requires keen attention, providing appropriate guidance without obstruction [15]. Glanceable interfaces are quick and convenient, but they suffer from three main drawbacks: (1) Limited Scalability: As the amount of AR content increases, Glanceable interfaces lose effectiveness due to the volume of data to be displayed and the potential to block real-world content [14][16]. (2) Viewing Angle: Problems can arise when viewing augmented content at an angle [17]. (3) Unintentional selection/interactions using gaze: This is the Midas Touch Problem (MTP) [18].

A Glanceable interface should be compact, enabling minimal information at a glance to avoid clutter, occlusion, and distraction [3]. When a large amount of data is necessary to guide the user through the task, AR elements occupy a large portion of the available visible space. Interactions with virtual content become less accurate and in workplaces such as machine shops, obstruction of the visual field is dangerous. Voice and gesture commands have been offered as an alternative to visual instruction. In a machine shop, however, AR interaction based on speech is not well-suited due to the high levels of noise, while mid-air hand gestures and wide controller motions can pose safety concerns [19].

In the first part of our work, we present an innovative desktop-based CNC machining training procedure, created as a serious game for CNC manufacturing training. The proposed serious game prepares the trainee in writing G-code and setting up virtual machines for completing milling and turning tasks. The serious game works as a standalone training system without the need for a supervisor or trainer to be present. The serious fame is focused on G-code training and basic machine setup as communicated by an engaging 3D environment and a streamlined training process. The serious game was developed in collaboration with the Micromachining and Manufacturing Modeling Lab (M3 lab) of the School of Production Engineering an Management of the Technical University of Crete. Technical development of the Serious game was accomplished by Grigorios Daskalogrigorakis and Salva Kirakosian. The M3 lab provided the 3D object models, 2D documentation in addition to theoretical guidance regarding manufacturing processes and manufacturing education. Salva Kirakosian was responsible for 2D and 3D design for the 3D virtual machine shop, while Grigorios Daskalogrigorakis was the gameplay designer and developer regarding the Gamification of the manufacturing process.

In the second part, we present Holo-box, a novel AR interface that combines Glanceable AR and world-based 3D interfaces employing a combination of eye-gaze and hand interactions. We evaluate this in a machine shop taking workplace safety considerations into account, boosting productivity and accuracy in a manufacturing process. Holo-box utilizes a 3-Level-of-Detail (3-LOD) architecture providing varying levels of information to the user while performing tasks in the machine shop. Levels can be expanded at will, through glance, to reveal additional information (see Fig. 1). The Glanceable interfaces enable rapid information retrieval with minimal real-world occlusion. Incorporating an AR 3D interface enabled with hand-tracking allows for complex, accurate interactions in a controlled environment. Holo-box uses custom eye interaction logic to automate accurate and natural direct eye interaction through blinking. The interface is deliberately designed to accurately decode gaze, even when the accuracy of the eye tracker is low.

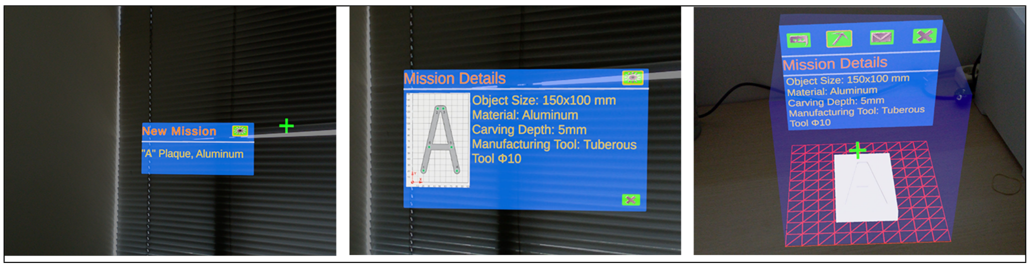


Figure 1: The proposed 3-LOD AR interface system.

Our work has the following contributions:

- We present Holo-Box, a novel 3-LOD interface system for AR combining both Glanceable and 3D interfaces [3]. Two initial LOD interfaces are controlled through glance (Glanceable interfaces), while the deepest level of LOD is a 3D world-based interface. The interface enables the user to view simple guidance information through the Glanceable interfaces or select to view the 3D interface to examine more detailed information.

- Holo-Box utilises a novel interaction system that utilizes a combination of eye-gaze blinking with a delay (dubbed blink-delay) for interacting with Glanceable interfaces to minimize unintended interactions. Objects remain selected and interactable even if the eye cursor exits their activation trigger unless another interactable object is selected. Glanceable interfaces are also visually adjusted to minimize errors in interaction, deliberately placing buttons on opposite sides of the interface. The 3D interfaces also allow hand tracking interactions with a delay (dubbed hand-delay). Users can engage interchangeably between hand-delay and blink-delay interactions; thus, interactable objects can be placed on any layout without restrictions [1].

- In the Machine shop Serious Game, we present a streamlined gamified presentation for manufacturing missions for milling and turning tasks [2]. The Serious Game was used for G-code education and received positive feedback from students. In addition, our proposed gamified presentation was adapted to AR for use in Holo-box to provide feedback on the real world [3]. The content shown in each LOD is adjusted so that each interface can be used to guide users with different levels of knowledge on its own. In the most compact Glanceable interface, we show only textual information, including the name and material of a given task. The second LOD level of the Glanceable interface offers additional text and 2D images. The subsequent 3D interface includes 3D objects and multiple interactable buttons. The Glanceable interfaces provide adequate guidance for experienced users, while the 3D interface provides additional information for inexperienced users, using 3D models relevant, in our case, to manufacturing tasks.

- We evaluated our work in Holo-box two times. In our first evaluation we found the issues of Holo-box and the limitations of our hardware and tracking, which were corrected for our final implementation [3]. In our second evaluation, we evaluate Holo-box by employing an object selection task in a pilot manufacturing scenario. The content displayed on all interfaces is used in presenting milling and turning manufacturing tasks in the form of gamified missions. Experiment results show that users complete missions faster and use more compact interfaces as they gain more experience. Our proposed interaction system compensates for the eye tracker's inaccuracies resulting in higher perceived accuracy on blink-delay inputs than hand-delay.

Chapter 2 includes a low level analysis of Augmented reality including theoretical knowledge, analysis of the most well known AR Head-Mounted Displays (HMDs) and the various ways to develop AR applications. In addition, chapter 2 also includes a brief analysis of real world Machine shop manufacturing, the new Industry 4.0 model and the necessity in including AR into the manufacturing workspace.

Chapter 3 provides an analysis of previous work including the use of gamified environments in education and/or training and the use of virtual environments and AR in workspace education. Chapter 3 also covers related work regarding AR interface types used in multiple scenarios, including adaptive and Glanceable interfaces.

Chapter 4 covers the implementation of a Serious Game for manufacturing education in which we designed a gamified environment for machine shop education, which served as the core of our AR application. The Serious Game was developed for remote G-code education, without the need for a supervisor. In this game, we designed a gamified manufacturing mission presentation, which was also adapted for use in the real world using AR.

Chapter 5 covers the first part of our AR implementation which includes our proposed 3-LOD interface system. In this chapter, we discovered the issues and limitations of our first attempt, and we analyze the way to overcome them.

Chapter 6 covers the full implementation of our work, which extends the previous chapter and solves the issues present in the prototype with Chapter 7 analysing our work in a low level inside the Unity Engine. Chapter 8 covers the user study used to evaluate our work.

2 Augmented Reality and the Industry 4.0 model

2.1 What is Augmented Reality

Augmented Reality is the act of superimposing digital artifacts on real environments [20]. In the Reality-Virtuality (RV) continuum, AR is a part of the broader Mixed Reality spectrum. In contrast to Virtual Reality (VR) where the user is immersed in a synthetic environment, AR aims to supplement reality with virtual objects.

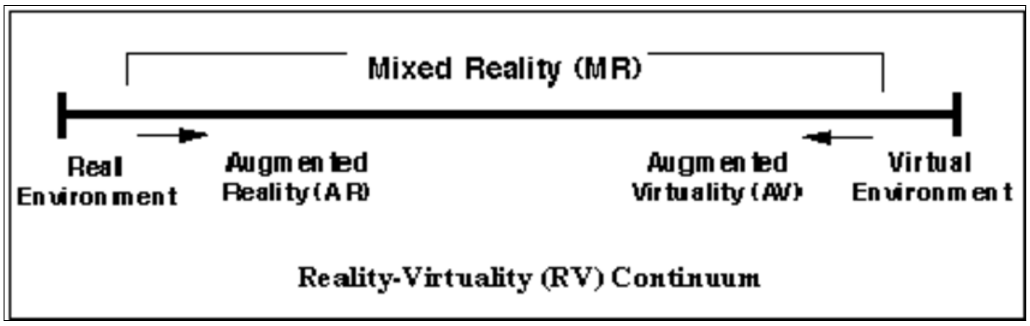


Figure 2: The Reality-Virtuality continuum [20]

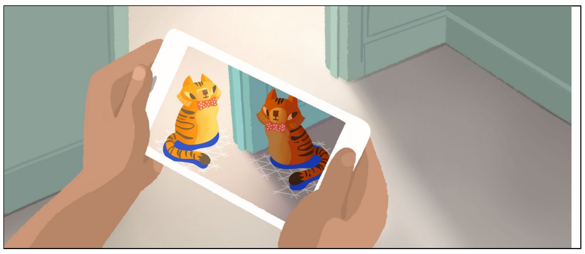
While early research limited AR in a way that required the use of Head-Mounted Displays (HMDs), later studies [21] differentiated the definition from the required technologies and instead defined any system that:

- Combines virtual and real
- Registers (aligns) virtual and real objects with each other
- Runs interactively in three dimensions and in real time.

2.2 Technologies of AR

AR technologies can be separated in three broad categories including:

- Video See-Through (VST) AR (Fig. 3a)
- Optical See-Through (OST) AR (Fig. 3b)
- Diminished Reality (DR) (Fig. 4)


(a)

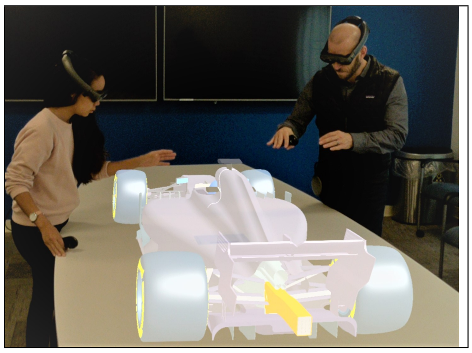

(b)

Figure 3: AR Display types. (a) Video See-through. (b) Optical See-through.

Video See-Through AR includes all types of AR that project the augmented world on a virtual screen (see Fig. 3a). The real world is captured on video through a camera, often residing on the opposite side of the screen. The video feed is then augmented with virtual objects and is shown on a screen to the user. VST is most commonly used on mobile-based AR or when using VR HMDs with front-facing cameras.

VST AR often uses Machine Vision (MV) to extract information regarding the real world and the user's movement inside it. In simple cases, only MV information is adequate for simple AR tasks allowing for cheap and rapid AR with minimal hardware requirements. In more demanding applications, MV is combined with readings from other sensors to provide more accurate localisation. Smartphone-based AR often combines ML with the built-in Inertia Measurement Units (IMUs) built into modern smartphones including gyroscopes, accelerometers, compasses and GPS for precise localisation, while the smartphone's touch screen is used for inputs when interacting with virtual objects. Other VST systems such as VR masks may also use external sensors for positional tracking, while most VR HMDs are paired with their own controllers for user inputs.

Optical See-through AR uses transparent prismatic displays to project holograms in front of the users (see Fig. 3b). Such displays are often mounted on a skeleton and worn like glasses and as such they are also known as "smart glasses" or AR HMDs.

OST AR allow the user to view the real world as is while only superimposing virtual objects. The HMDs are often equipped with a plethora of sensors specifically tailored to the needs of AR, most commonly for Simultaneous Localisation And Mapping (SLAM) or user interactions. There are multiple available AR HMDs available each with their own hardware and Software Development kits (SDKs) available which are tailored towards commercial or academic use (see Chapter 2.3). Interactions in OST AR HMDs often use a combination of interaction types which may include controllers, hand tracking, speech recognition as well as eye tracking among others based on the hardware used.

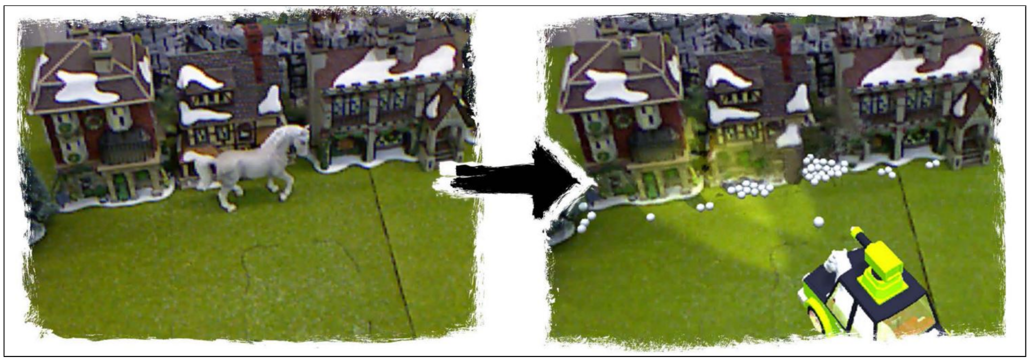


Figure 4: Diminished Reality as seen in [22]

Diminishing Reality is an alternative approach to traditional AR technologies. Both VST and OST AR are considered "additive AR" in the sense that they take the real world's geometry as input and then "add" virtual objects to create the augmented environment. DR takes the real world as input and then attempts to recreate a new real world view after removing certain real world objects [22].

Unlike VST and OST which focus on adding virtual objects and then adjust their positions and visuals to make them blend into the real world, DR focuses more on world reconstruction techniques to recreate parts of the real world that were obstructed by the removed objects. This is often accomplished through the use of Machine Learning, external sensors that detect the real world through other viewpoints or pre-scanned world reconstruction information.

DR is the least developed AR technology of the three with practical studies only showing after the recent increase in ML technologies in the last decade. Currently there is no mainstream hardware or software for DR applications outside of prototype ML reconstruction libraries.

In this work, we focus our efforts in OST techniques using AR HMDs. Our goal is to design a Glanceable interface system that can be used while performing real world tasks, while our primary concern is to preserve the real world view for safety reasons. Smartphone-based AR requires occupying the user's hands which is not ideal for performing real world tasks while VST HMDs increase the risk of injury or simulation sickness present in the delay between moving and the video feed on the display catching up to the real world. Some techniques of DR would be useful such as viewing obscured real world objects for providing additional safety but since there is minimal documentation on how to achieve this and our primary goal is guidance and not spatial awareness this is left as an extension for future work.

2.3 Augmented Reality Head-Mounted Displays (HMDs)

Following is the hardware analysis of the AR HMDs we considered for our application and a comparison between them.

2.3.1 Microsoft Hololens

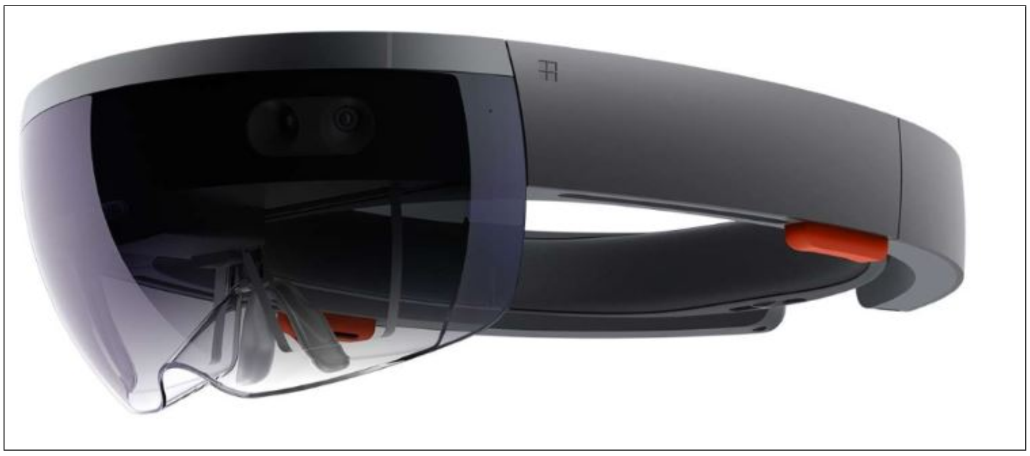


Figure 5: Microsoft Hololens

Released in 2016 for a price of approximately 5.000,00 dollars (for developers the price was reduced to 3.000,00 dollars), the Hololens's announcement in 2015 sparked many companies to hasten their development in the field of AR HMDs, like Sony (unnamed AR HMD prototype), Magic Leap (Magic Leap One) and DAQRI (DAQRI AR HMD) as well as smartphone-based AR like Google (ARCore) and Apple (ARKIt), sparking a large release of AR-based hardware and software in 2016 and 2017.

With the release of Windows phones, the Xbox One and Windows 10, Microsoft had also announced the Universal Windows system, and with the announcement of the Hololens, Microsoft included it in the Universal Windows system, promising all applications developed for this new system to work in desktop, smartphone and AR environments.

The Hololens is a lightweight head worn mask with the bulk of the sensors on the front side of the mask and the processing units positioned around the head to counterbalance the weight. The Hololens included an Intel Atom 4 core 32-bit CPU and 2GB RAM as well as a custom-made Holographic Processing Unit (HPU) which is a CPU specifically made for spatial awareness and location processing to reduce the load in the main CPU. It also included two AR displays with 34 degrees diagonal FoV with a resolution of 1280x720 pixels at 60Hz refresh rate. The low specs of the Hololens compared to contestants were especially to preserve the battery life, which lasts up to 3 hours of use.

The Hololens uses a single Inertial Measurement Unit (IMU) for positional tracking, four environmental understanding cameras and one depth camera for world mapping, one camera for video recording and streaming to any Windows 10 or Universal Windows device as well as four microphones for speech detection.

The Hololens supports three types of inputs. The first is a "clicker", a controller with a single button that can be used for interactions. The clicker does not contain any form of tracking or precise inputs other than the single button. The second is speech recognition. By incorporating Microsoft's Cortana allowed for speech recognition and use of simple voice commands like "select" "launch" or "shut down". As an alternative, users could use natural gestures like a "tap" or "bloom" motion to interact with objects, although the Hololens only detected the gestures without tracking the hand positions at all and the HMD did not distinguish between the hands, offering a single-hand interaction system.

The Hololens uses a gaze-based interaction system. The user focuses the center of their viewpoint on the desired object, which is considered selected and then presses the clicker, performs a gesture or speaks the voice command to interact with the virtual object. After user feedback and comparison to other system it was later proven this interaction method was exhausting for long-term usage.

2.3.2 Microsoft Hololens 2

Figure 6: Microsoft Hololens 2

Announced to be released in Summer 2020 for a price of 3.500,00 dollars, the Hololens 2 is a bulkier system that improves on everything the first one lacked. The resolution increased to 2048x1080 pixels per eye and the FoV increased to 52 degrees diagonal, resulting in the same pixel density as the first one. The CPU changed to a Qualcomm Snapdragon 850 octa-core one and the HPU was enhanced to the HPU version 2, although Microsoft has not given details of how it was enhanced. The RAM was also increased to 8GB making the specifications of the Hololens v2 comparable to high-end smartphones. Although the battery capacity increased so did the power consumption, and the Hololens 2 has the same battery life as the first one (approx. 3 hours).

The Hololens 2 also includes eye tracking, eye position-based depth adjustment and iris scanning for user profiling. The microphones also changed to a five-microphone array for surround-based audio detection. The depth sensor is also replaced with a Kinect Azure sensor, with all the new software and hardware capabilities developed for this new system.

The OS is also changed from the Universal Windows system to the new Windows Holographic OS which is based on the Windows 10 OS but has additional functionality required for AR systems, such as an improved interaction system using gestures and hand tracing. The gesture detection system was changed to a two hand tracking system with precise hand position tracking and even more gestures, allowing for touch-based interactions instead of gaze-based ones.

2.3.3 Magic Leap One

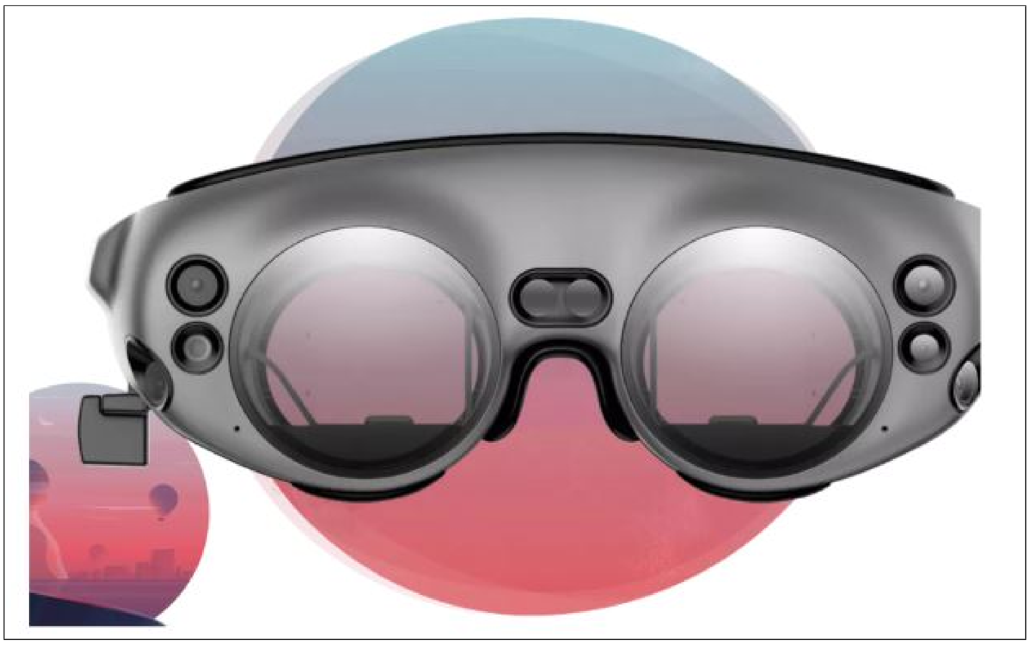


Figure 7: Magic Leap One

Magic Leap One is the first AR HMD of the Magic Leap company. It is a standalone AR HMD for complete optical see-through experiences without the need for a PC. The HMD consists of the HMD, the Magic Leap Lightpack and the Magic Leap Control. The Magic Leap was released as a prototype for non-commercial use in August 2018 for a price of 3.000,00 dollars.

The Magic Leap contains two wide FoV, for AR standards, see-through lenses that allow for up to 50 degrees diagonal FoV with a resolution of 1280x960 pixels at 120Hz for each eye. The HMD also contains two infrared depth sensors (center of the HMD), two cameras for key point detection (lower cameras on the sides of the HMD) as well as two additional cameras for video recording and video streaming to a PC (upper cameras on the sides of the HMD). The optical lens also contains three infrared sensors used for eye tracking and a Bluetooth sensor (”box” at the bottom left of the HMD) that connects to the Magic Leap Control.

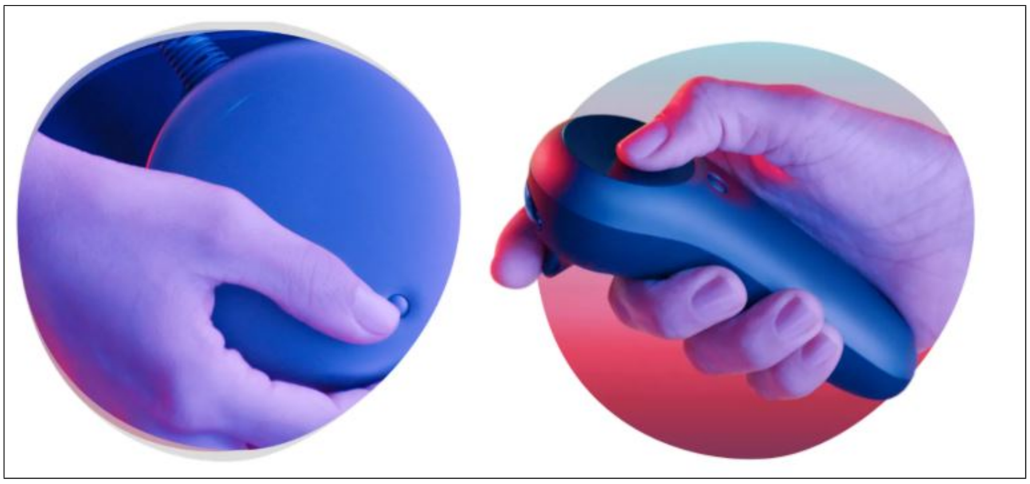


Figure 8: Magic Leap Lightpack and Magic Leap Control

The Lightpack is a pouch worn on the belt or pocket that contains all of the processing power and batteries of the device, which is wired to the HMD at all times. This makes the HMD itself significantly lighter than others, like the Hololens, and also allows for a larger battery, allowing for over 4 hours of autonomous usage. The Lightpack contains a custom CPU that includes 2 Denver 2.0 cores and 4 ARM Cortex A57 cores (1 Denver and 2 ARM cores available to applications) as well as an NVIDIA Pascal GPU with 356 CUDA cores as well as 8GB of RAM (4GB are available to applications). The Lightpack uses an Android-based OS named LuminOS, which is custom-made and adapted for AR.

The Magic Leap control is a controller with 6-DOF (position and orientation) tracking capabilities, two buttons, one trigger and a circular touchpad. The Magic Leap Control is connected to the HMD through a dedicated sensor and actual tracking calculations are done inside the HMD allowing for larger battery life of the controller (approx. 8 hours) and cross-referencing of internal tracking with the visual tracking of the HMD.

The Magic Leap SDK allows programmers to use various data extracted from the sensors in a high-level interface. Through the use of a Simultaneous Localisation And Mapping (SLAM) system, the Magic leap constantly scans the real world and tracks the HMD's position inside the scanned space. The device re-creates an untextured mesh of the user's room. Programmers can use this virtual mesh to make augmented content spatially aware, for example attaching interfaces to walls or allowing virtual characters to walk on flat horizontal surfaces. With the update released in May 2020, these mesh rooms can also be uploaded to the cloud, shared between different HMDs and multiple HMDs can be used for collaborative scanning.

In addition to world scanning, the HMD also tracks the user's hands when they are within vision of the cameras. The SDK generates 15 points tracked for each hand in 3D coordinates (2-3 points per finger, palm and wrist) even when some points are partially obscured. The HMD also provides eye tracking data in two forms: the rotation of each eye as well as the 3D point in the world where the user gazes at.

Finally, the Magic Leap HMD automatically detects some predefined gestures for each hand separately. These gestures can be used as an alternative interaction method to traditional controllers. Since the gestures are tracked along the world scanning, the automated gesture detection works at 120Hz for seemingly instantaneous detection. In addition, Magic leap supports voice recognition, speech detection and text-to-speech, but these functionalities are left up to the developers to use as needed, the Magic Leap itself does not use or post process this data.

2.3.4 AR HMD comparison

For the purposes of our application we compared the three aforementioned HMDs to determine the best one for our application. The Microsoft Hololens was the most compact of the three and also the most comfortable for long sequences of use but the limitation of interacting with the center of the user's gaze and a clicker as well as the smaller FoV made for a less than ideal experience.

The Microsoft Hololens 2 and the Magic Leap One are similar in terms of theoretical interaction methods and FoV. In practice, the Hololens 2 proved to recognise speech commands significantly faster and with higher accuracy, while the Magic Leap One proved more accurate in hand tracking and recognised gestures with minimal latency while both HMDs provided similar eye tracking performance (approx. 5cm offset on a blank wall 1m away with ideal lighting conditions) with high flickering that often resulted in misinteractions. In addition, both HMDs were prone to heating at uncomfortable temperatures on the user's forehead above the prisms after the first hour of use. Excluding the heating, the Magic Leap One HMD proved to be better for long usage periods due to the Magic Leap Lightpack being worn on the hip and not the head and thus having less weight and more stability on the user's head.

Ultimately, the major drawback of the Hololens 2 was the documentation. The Magic Leap One has extended documentation and tutorials with backwards compatibility for all available libraries and SDK versions. The Hololens 2 on the other hand uses the same websites for documentation and forums as the first Hololens while the two HMDs have different SDKs and thus resulting in many pages having wrong or no documentation while searching the forums often results in outdated answers that are no longer supported.

2.4 Application development in Augmented Reality

With the release of Hololens back in 2016 and onward, AR HMD application development has been streamlined across most commercial AR HMDs. App development can be done through the use of each of the following tools:

- Native development through Application Programming Interfaces (APIs)
- Cross-platform development through external tools
- Third party Game Engine Packages

In the following chapters we will analyze the use of these tools specifically for developing apps for the Magic Leap One which we use in this work, with similar tools being available for other HMDs such as the Hololens and Hololens 2.

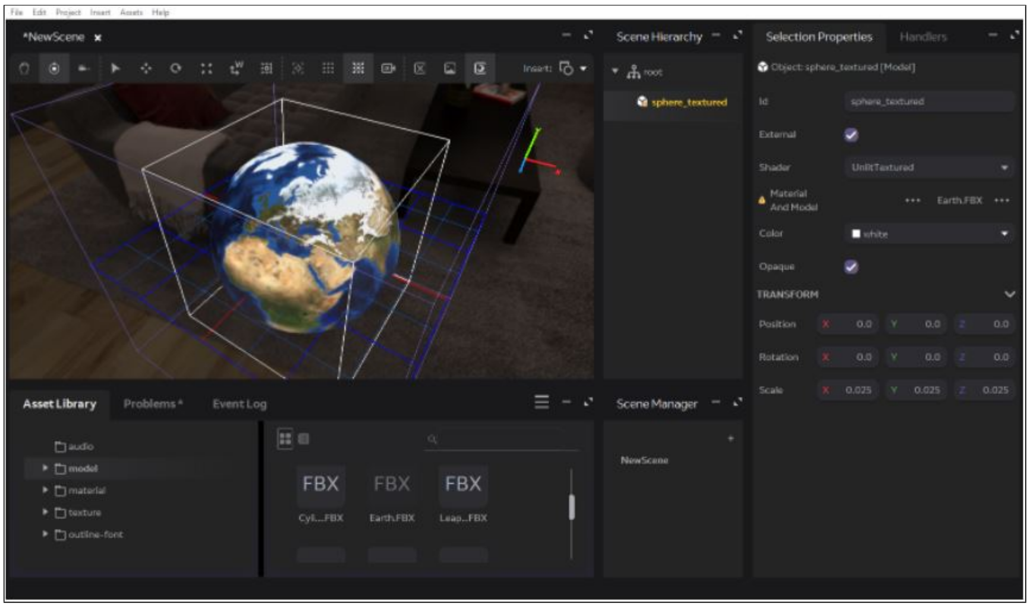
2.4.1 Native platform Development

All AR HMDs similar to other hardware systems support native build using their own libraries which is usually comprised of low level code with direct access to the system's features through OS calls. Such systems are ideal for prototyping but they require a high degree of specialisation in coding for a specific platform and is not ideal for non-programmers or amateurs.

Magic Leap One supports two forms of native development: a C API and the Lumin Runtime Framework.

C API. The Magic Leap One's OS, LuminOS is a Linux-based OS written in C. The developer API consists of libraries that allow other code to connect with the core OS. The development API is available for both C and C++ coding language native programming.

Lumin Runtime Framework. This framework provides a predefined code structure for designing apps of two available types including Landscape and Immersive. Landscape apps can run and render simultaneously with other landscape apps and are contained withing a volume called a Prism that is placed onto the real world. Immersive apps suspend other apps during runtime and utilise the entire available space.



Object: sphere_textured [Model]

Id	sphere_textured
External	✓
Shader	UnlitTextured
Material And Model	Earth.FBX
Color	white
Opaque	✓

TRANSFORM

	X	Y	Z
Position	0.0	0.0	0.0
Rotation	0.0	0.0	0.0
Scale	0.025	0.025	0.025

Figure 9: The Lumin Runtime Editor using the UI kit

The Lumin Runtime Framework works on a server/client architecture where each application sends server calls to the OS allowing for multiple applications to run at once as well as application sharing between multiple users. The Runtime Framework includes APIs that allow for interacting with the following:

- The scene graph - a spatial and hierarchical structure representing the scene
- Transforms that manipulate the spatial properties of scene graph nodes and tween animations
- 3D models with material and skeletal 3D model animations
- 2D sprite animation

- High fidelity spatialized text rendering for spatial computing
- 2D and Soundfield Audio
- 2D and stereoscopic video
- Rigid Body collision physics
- Real time particle FX
- Input events and haptics control
- Hand gestures
- Speech to text
- Real time spatial computing technology for rendering objects, lighting and shadows, and real world occlusion

The Lumin Runtime environment is available for use through Visual Studio, Visual Studio Code and the Lumin Runtime Editor (Fig. 9). In addition, Magic Leap provides the Magic Leap UI kit for quickly designing ready to use apps with premade UI elements with consistent colors, shapes, sounds and animations that include:

- Labels
- Buttons
- Toggles
- Sliders
- Images
- Text Boxes
- List Boxes
- Drop-Down Lists
- Default Hover states
- Rect, Linear and Grid Layouts
- Scroll Views
- An Event system (hover, click, focus etc.)
- Cursor interaction with the Magic Leap Control or Mobile App.

2.4.2 Cross-platform development

Most applications nowadays are developed for a variety of platforms. As such developers tend to use higher level libraries that allow for developing the application once and then building them seamlessly to a variety of platforms. Magic Leap includes two web-based approaches to app development that allow for cross-building with other systems. These include MagicScript and Web Platform development

MagicScript is a framework for developing native, mixed-reality apps with JavaScript to run on Lumin OS and Magic Leap devices. Apps for Magic Leap written in MagicScript can be spatial computing Landscape or Immersive apps.

A React Native extension for MagicScript, called MagicScript Components, is available to create spatialized front-ends with React.js, declarative elements, and Javascript. MagicScript Components allows the same source code to be built and run on Lumin OS, iOS, and Android. The initial Get Started in MagicScript section introduces MagicScript Component apps, using the simplicity of the framework.

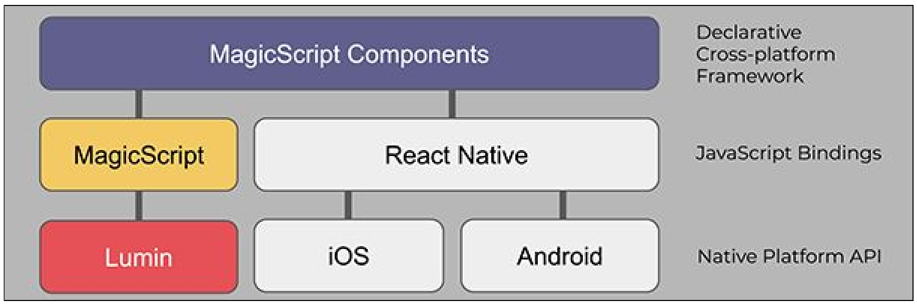


Figure 10: MagicScript cross-platform architecture for Magic Leap and Mobile development

MagicScript uses NPM (Node Package Manager) to assemble apps from multiple components (NPM packages). This supports the use of external NPM packages and frameworks in MagicScript apps. MagicScript apps can be extended with open source JavaScript libraries such as Matrix.js for transform math, Lodash for vector operations, and Xeogl and Three.js for 3D rendering.

MagicScript enables support for WebGL for rendering Quad nodes using Planar resources in Landscape and Immersive applications. MagicScript supports the most common APIs of WebGL 1.0 and a subset of functions from WebGL 2.0.

Web Platform development allows programmers to develop Magic Leap applications in JavaScript, HTML or CSS. Web developers working with Magic Leap web platforms can create interactive sites and apps that take advantage of the full spectrum of spatial computing.

Web-based development allows developers to create a vast array of AR experiences from simply extending a website in AR to creating full fledged AR experiences with only a few lines of code. Magic Leap is embracing the W3C's immersive WebXR standards as we transform how humans interact with data and content.

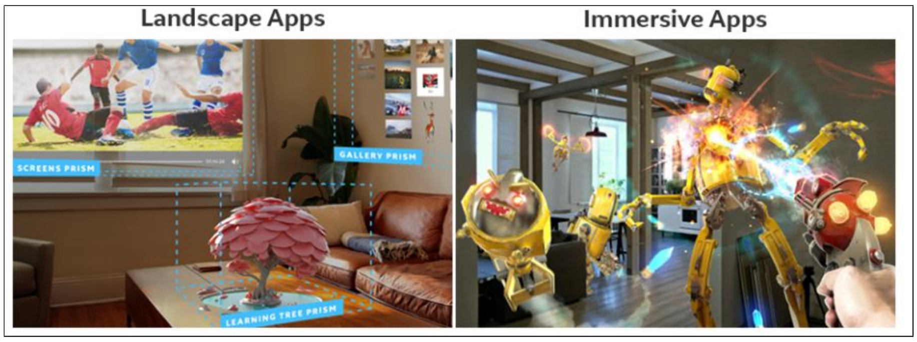


Figure 11: Magic Leap Web based Apps

Magic Leap apps can be created in one of two modes: Landscape and Immersive. Landscape apps work similar to a desktop experience with multiple apps that appear inside three-dimensional prisms like windows on a desktop environment. Landscape apps can open other apps using deep links, Web View, and OAuth calls. Immersive apps on the other hand take over the user's experience by utilising the whole viewing area, comparable to a phone app or a console game.

The Helio Web Platform is a web browser available for the Magic Leap One that extends any 2D browsing experience to spatial computing. An easy to use HTML-declarative library allows programmers to code interactive spatial computing content with only a few lines of code. Programmers can define the size and source of a 3D model to appear in Helio with a single line of code and from there programmers can animate, transform content, allow for user input and control content behavior as they would with any other web-based system.

Landscape applications can be developed in one of three available modes including Detached CSS, Prismatic or Progressive web apps while Immersive apps are designed using the WebXR API.

Detached CSS works similar to responsive web pages, which adjust to scale in order to optimize the viewing experience on any device from phones to wide screen monitors. It allows programmers to modularize the web experience by adding CSS attributes to any element in a page to detach that element and moving it elsewhere taking all its CSS properties with it. For example, a museum experience with interactive animated objects can be designed by pulling multiple feeds from a website and placing them on walls, tables and desktops.

Prismatic is an open source declarative JavaScript library that lets programmers add 3D models and define a 3D space viewable on Helio and non-Helio browsers. By using simple HTML tags with inline attributes and CSS styling, 3D content that pops of the page can be rendered. Users can grab and place the prismatic app anywhere on their environment. Prismatic lets programmers animate, transform, and program interactions with web content.

Progressive Web Apps allow designing a web page and then wrapping it as an app. Web app APIs let things run in the background once they're downloaded. This allows programmers to open something that looks more like an app and less like a web page, but gives the Native experience. Progressive web apps lets programmers publish web content to Magic Leap devices, streamlining distribution and monetization. Enterprises that want to restrict internal apps can distribute web apps to devices without publishing them on the internet.

WebXR is a web-based device API that lets programmers develop immersive, WebGL-based apps that users launch from Helio that then encompass the user's entire viewing space. WebXR is an evolving standard driven by the World Wide Web Consortium (W3C) that aims to bring augmented and virtual reality to a common set of guidelines. An example of such an app would allow users to scroll through a catalog of 3D models, then select a model to enter an immersive app for a closer view of the model. When they close the immersive view they return to the web page experience. Video & multimedia opportunities include communication experiences with avatars in an immersive world or gaming experiences with interactive components.

WebGL development is currently complex, but Three.js simplifies JavaScript WebGL, making it easier to create elements, import shaders, and otherwise develop complex experiences. On top of Three.js, Babylon.js and Aframe are available to make Three.js declarative.

2.4.3 Game Engines

A game engine is a software-development environment designed for people to build video games. Developers use game engines to construct games for consoles, mobile devices, and personal computers. The core functionality typically provided by a game engine includes a rendering engine ("renderer") for 2D or 3D graphics, a physics engine or collision detection (and collision response), sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, localization support, scene graph, and may include video support for cinematics. Implementers often economize on the process of game development by reusing/adapting, in large part, the same game engine to produce different games or to aid in porting games to multiple platforms.

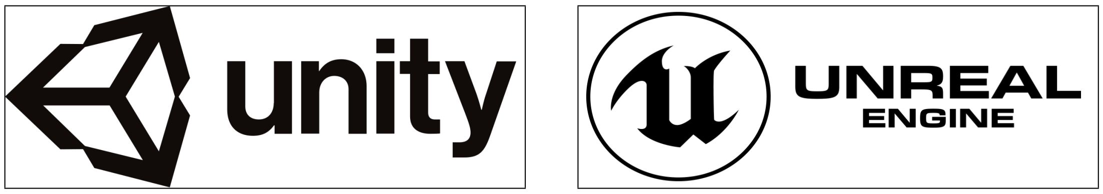
In many cases game engines provide a suite of visual development tools in addition to reusable software components. These tools are generally provided in an integrated development environment to enable simplified, rapid development of games in a data-driven manner. Game engine developers attempt to "pre-invent the wheel" by developing robust software suites which include many elements a game developer may need to build a game. Most game engine suites provide facilities that ease development, such as graphics, sound, physics and AI functions. These game engines are sometimes called "middleware" because, as with the business sense of the term, they provide a flexible and reusable software platform which provides all the core functionality needed, right out of the box, to develop a game application while reducing costs, complexities, and time-to-market — all critical factors in the highly competitive video game industry.

Like other types of middleware, game engines usually provide platform abstraction, allowing the same game to be run on various platforms including game consoles and personal computers with few, if any, changes made to the game source code. Often, game engines are designed with a component-based architecture that allows specific systems in the engine to be replaced or extended with more specialized (and often more expensive) game middleware components. Some game engines are designed as a series of loosely connected game middleware components that can be selectively combined to create a custom engine, instead of the more common approach of extending or customizing a flexible integrated product. However extensibility is achieved, it remains a high priority for game engines due to the wide variety of uses for which they are applied. Despite the specificity of the name, game engines are often used for other kinds of interactive applications with real-time graphical needs such as marketing demos, architectural visualizations, training simulations, and modeling environments.

With the rapid improvements in technologies for cross-platform builds in the gaming industry, the use of high level middleware called "Game Engines" has become the norm. A game engine is used to automate linking all the subsystems included in a game and provide a Graphical representation of the game. The most widely used game engines for AR development are the Unity Engine [23] developed by Unity Technologies and Unreal Engine [24] developed by Epic Games. Both Unity and Unreal are free to use for noncommercial and academic use.

Unreal Engine - The Unreal Engine is a game engine developed by Epic Games, first showcased in the 1998 first-person shooter game Unreal. Although initially developed for first-person shooters, it has been successfully used in a variety of other genres, including stealth, fighting games, MMORPGs, and other RPGs. With its code written in C++, the Unreal Engine features a high degree of portability and is a tool used by many game developers today, with it being source-available.

Unreal engine is developed in C++ which is a lower level language than Unity's C#, allowing for better performance in high-end games. Unreal engine favors more flexibility for the programmers and allows for more direct connection with external libraries and hardware optimisation in favor of less intuitive cross-platform development compared to Unity.



(a) Unity (b) Unreal Engine

Figure 12: Widely known Game Engines

Unity - Unity is a cross-platform game engine developed by Unity Technologies, first announced and released in June 2005 at Apple Inc.'s Worldwide Developers Conference as a Mac OS X-exclusive game engine. As of 2018, the engine had been extended to support more than 25 platforms. The engine can be used to create three-dimensional, two-dimensional, virtual reality, and augmented reality games, as well as simulations and other experiences. The engine has been adopted by industries outside video gaming, such as film, automotive, architecture, engineering and construction.

Unity gives users the ability to create games and experiences in both 2D and 3D, and the engine offers a primary scripting API in C#, for both the Unity editor in the form of plugins, and games themselves, as well as drag and drop functionality. Prior to C# being the primary programming language used for the engine, it previously supported Boo, which was removed in the Unity 5 release, and a version of JavaScript called UnityScript, which was deprecated in August 2017 after the release of Unity 2017.1 in favor of C#.

The engine has support for the following graphics APIs: Direct3D on Windows and Xbox One; OpenGL on Linux, macOS, and Windows; OpenGL ES on Android and iOS; WebGL on the web; and proprietary APIs on the video game consoles. Additionally, Unity supports the low-level APIs Metal on iOS and macOS and Vulkan on Android, Linux, and Windows, as well as Direct3D 12 on Windows and Xbox One.

Within 2D games, Unity allows importation of sprites and an advanced 2D world renderer. For 3D games, Unity allows specification of texture compression, mipmaps, and resolution settings for each platform that the game engine supports, and provides support for bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects.

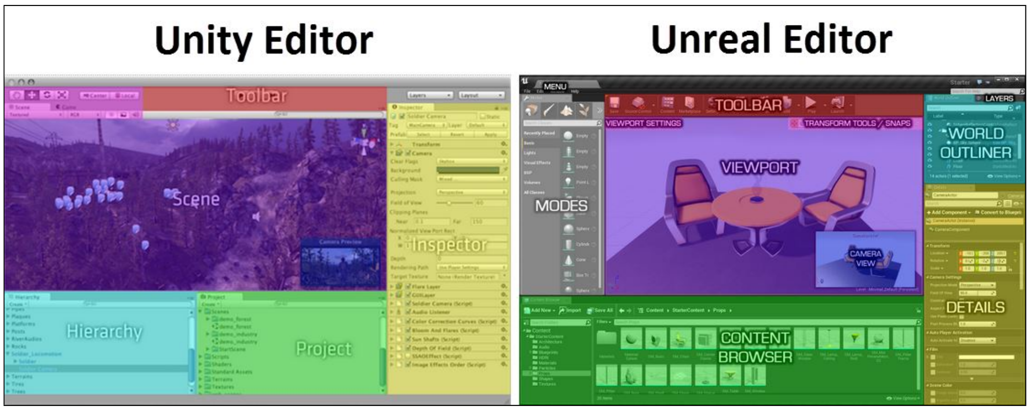


Figure 13: Unity Editor and Unreal Editor

The Magic Leap Unity and Unreal Packages allows for designing Magic Leap applications through the Unity or the Unreal editor. Applications are designed similar to any other desktop, mobile or console game through the engine's tools while the packages allow for connection with the hardware's resources, external tools and automate the building process into native code. By using a game engine programmers can build Magic Leap apps while utilizing a vast library of tools readily available inside the game engines inside a streamlined graphical environment, allowing for the development of applications of significantly higher quality with a low amount of effort compared to native building them.

Now-days that new consoles and devices are being made, Unity seems to be supporting almost every new product. This is a huge advantage since using Unity allows developers to release on platforms that other engines don't support. Unity's C# is a high-level programming language that significantly simplifies the coding process by automating multiple background processes, such as garbage collection, rendering calls and hardware callbacks, through the engine's libraries.

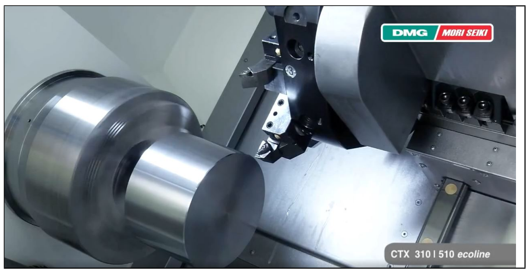
Unreal Engine seems to be a better option for games needing graphically intensive performance and for companies that have very experienced development teams. However, for small companies looking to target multiple platforms and monetize effectively, as a result more companies prefer using Unity.

In this work we will exclusively work through the Unity editor using the Magic Leap Unity Package which is available on the Magic Leap Developer Github https://github.com/magicleap/MagicLeapUnityExamples. Unity supports stable framerates for a wide range of devices with varying hardware capabilities. The ease of development using C# is also preferred over optimising low level operations in C++ resulting in faster development times. Unity's more streamlined coding has also resulted in a larger community of developers offering assistance in Unity's official forums, allowing for rapid feedback and well designed documentation.

2.5 Machine shop manufacturing

The manufacturing industry is a wide umbrella that encompasses many production facilities. Machine shops usually refers to small factories where machining, or subtractive manufacturing is done.

Subtractive manufacturing is defined as the process of taking a stock object, usually metal, plastic, glass or wood, and ”subtracting” material from the object to produce small mechanical parts such as gears, car replacement parts or even customized mechanical components for any use. Conversely, ”additive manufacturing” defines procedures that build objects by adding and shaping stock material,such as in 3D printing. Both subtractive and additive manufacturing operate using the same tools with different logic, in additive manufacturing we aim to build the desired object in layers from bottom to top while in subtractive to remove unwanted material from a predefined stock from top to bottom.



(a) Turning Machine

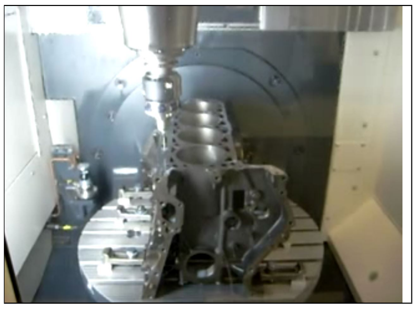

(b) Milling Machine

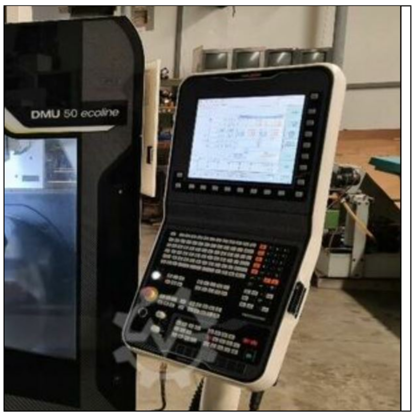
Figure 14: Examples of Manufacturing machines

Two of the most widely used machines used in Machine shops, and the two machines we target in this work include turning and milling machines.

Turning Machines (see Fig. 14a) consist of a cylindrical stock material which is spun rapidly while using a cutting tool that cuts the material evenly around its periphery while moving with two Degrees Of Freedom (2-DOF) either parallel or perpendicular to the material's central rotation axis.

Milling Machines (see Fig. 14b) on the other hand consist of a rectangular stock material attached to the machine's base with a rotating cutting tool that cuts the material from above. Milling machines can have up to 6-DOF movement including three-dimensional movement for the cutting tool inside the cutting chamber as well as three dimensional rotation for the material base.

Operating milling and turning machinery is commonly done through three common ways. The simplest way to operate a machine is through an attached Numerical Control (NC) unit (see Fig. 15a). Using an NC unit allows the operator to manually input commands to the machine one at a time and is ideal for simple commands or performing initial preparation before other complex processes.



(a) Numerical Control (NC) unit

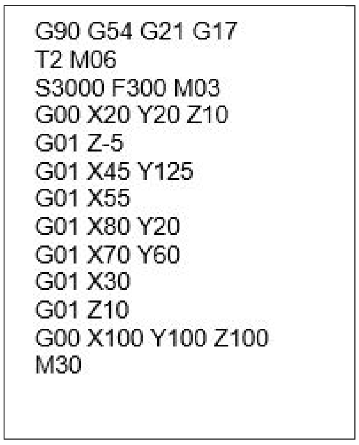

```
G90 G54 G21 G17
T2 M06
S3000 F300 M03
G00 X20 Y20 Z10
G01 Z-5
G01 X45 Y125
G01 X55
G01 X80 Y20
G01 X70 Y60
G01 X30
G01 Z10
G00 X100 Y100 Z100
M30
```
(b) Example G-code program

Figure 15: Manufacturing operation types

Complex operations on the other hand often require pre-programming the cutting procedure and loading it onto the machine at once. Manufacturing commands are written in G-code (see Fig. 70) which is a low level programming language that translates a single line of code into an appropriate motion of the cutting tool on its axis (e.g. G00 X20 Y20 Z10 translates to move on a straight line to the position X=20mm, Y=20mm, Z=10mm without cutting. The same command starting with G01 would translate to perform the motion while cutting meaning with a higher rotation speed and slower translation speed in order to perform the cut without breaking the tool). Finally, alternatively to writing the G-code of an operation a designer can opt to create a 3D model of the finished product in a designing program such as AutoCAD. Interpreters can be used to translate the 3D geometry into G-code commands for the manufacturing process. Although this is seemingly the simplest way of operating machines the use of G-code is preferred in most cases as it is simpler to learn for users without designing or programming knowledge.

2.6 The Industry 4.0 model

With the rapid advancement of new technologies, manufacturing is slowly aiming for the next stage in industrial revolution. The first and most widely known industrial revolution was achieved with the use of steam engines, followed by the second with the introduction of electricity and production lines for mass production and the third industrial revolution with the addition of computer interfaces inside the workspace.

The proposed Industry 4.0 extends on the computer-oriented approach of current manufacturing machine shops and introduces three new technologies for further automation of the manufacturing process [12]. In the proposed Industry 4.0, Artificial Intelligence (AI) is used for decision making regarding the operation of machinery, optimising the production lines or even monitoring and ordering human workers inside the factories. By utilising sensors inside the factory through Internet of Things (IoT) and cloud-based computing the manufacturing process can be automatically monitored and the AI can take more accurate decisions such as adjusting the working parameters of various machinery or even turning certain machinery off for reduced costs. The AI can also detect abnormalities and malfunctions on various machines and request technical assistance to repair or replace malfunctioning parts.

Although this new Industry standard significantly improves the manufacturing process, factories under this standard need significant redesigns in their structures to operate so transitioning to the Industry 4.0 model is still theoretical. Some major challenges to the transition to the Industry 4.0 model include [11]:

- *Lack of finding qualified personnel.* Current manufacturing personnel are not qualified to work using the new technologies.
- *Lack of advanced education systems for training personnel.* General education in most countries does not include the new technologies in the education curriculum.
- *Lack of readiness for innovation.* Supervising entities are not mentally prepared to risk investing in this new technology.
- *Lack of Information Technology infrastructures.* Current IT technologies are inadequate for efficient Human-Computer interactions in the new standard.

With this increase in automation the use of physical screens to monitor and operate each machine individually is no longer necessary. Instead, human operators can use AR as a machine interface, allowing an operator to monitor the whole factory from a single interface from anywhere inside the factory. An operator can use AR to not only operate a single machine but they can also view the manufacturing process in detail through the IoT sensors readings and by connecting directly to the operating AI to understand any malfunctions in the operating process.

In our work, we aim to introduce AR to the manufacturing machine shop by adding an intuitive interface system for rapid information retrieval. Our interface system can be used for improved guidance for inexperienced users, serving as a stepping stone into introducing new technologies into manufacturing. In addition, using guided AR allows trainers to educate new trainees on the real world instead of theoretical studies allowing for more hands-on experience during the training process.

3 Related Work

In this chapter we analyze various academic works related to our application. We start by explaining what Gamification is and then we present examples of gamified systems and what they offer versus their traditional non-gamified counterparts. Next, we showcase examples of virtual training environments used for industrial education developed for desktop systems or virtual reality and then we move to similar examples using augmented reality. We then compare VR and AR as technologies to clarify the perks to using each system in order to identify the strengths of using AR over other systems. Finally, we present examples of using AR in workplace guidance while we then focus on Glanceable AR which is the main focus of our work.

3.1 Gamification in training

3.1.1 System Gamification

Most games are designed to bring relaxation, focus shift, the sense of competition, yet, serious games are intended for a somewhat different purpose: to learn something or master a certain skill using a game's format. Often, serious games are built as simulators designed to practice a skill – flying an aircraft, administering first aid, driving a car, planning a city – all in a game's environment. In fact, serious games are not recent. Microsoft Flight Simulator was released as far back as in 1982. Of course, in 2019 its gameplay and design look a bit childish and naive, but the concept is the same. Today, however, we wrap this concept in such powerful technologies as augmented reality and virtual reality to create truly immersive serious games.

Serious games and Gamification are often confused, as they both mean using game techniques to achieve a non-entertainment purpose. At the same time, there is a difference. **Gamification** means using certain elements of a game in an otherwise ordinary process.

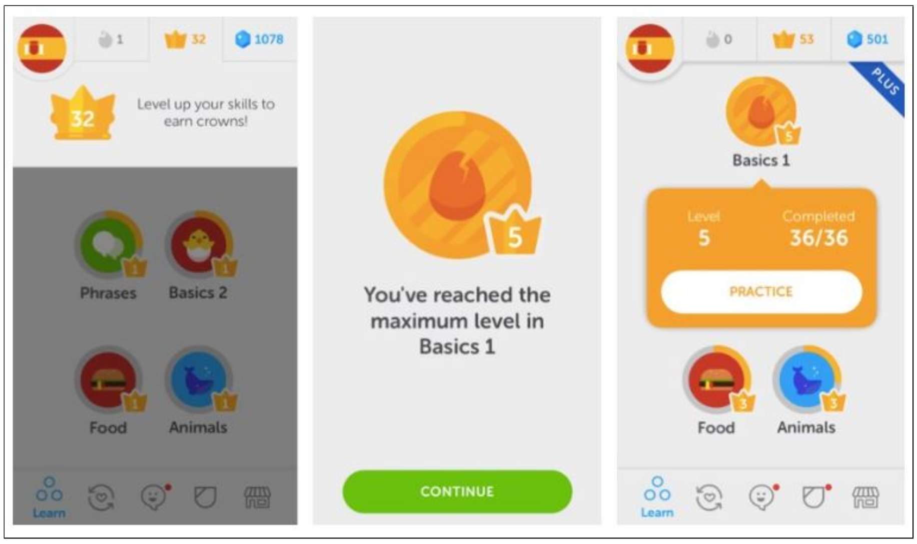
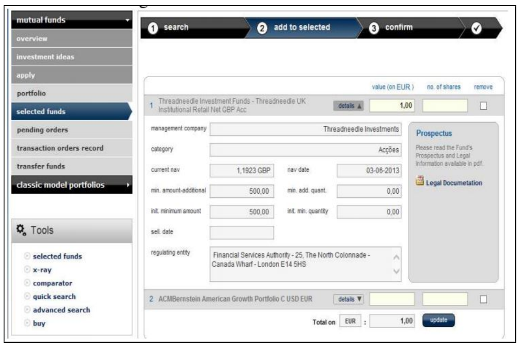


Figure 16: Duolingo language learning app [25]

As an example, introducing points, stars or badges as incentives in a learning process is Gamification. The learning process is not structured as a game, but you can get a trophy or a crown by reaching a certain level. A good example of the Gamification technique is Duolingo[25] (shown in figure 16), a language learning app. The learning process is not different from what you get at a language school or with a teacher – you memorize words, translate sentences, choose correct word forms. However, the app encourages you to earn crowns, compete with other users, win crystals to use towards in-game purchases. That's Gamification.

Now, in a **Serious Game**, the learning process itself is a game. The skills and knowledge are mastered by completing the game tasks. The game objectives may be different – from getting a physical skill, such as driving a car or operating a piece of machinery, to specific knowledge, for example, first aid techniques or management.

Rordrigues et al.'s study [26] examined the relationship between perceived socialness, perceived ease-of-use, perceived usefulness, perceived enjoyment and intention towards using an e-banking mutual funds Gamified application (see figure 17). Results show that customers that use the Gamified application perceived that the ease-of-use has a large positive influence on the intention to use the application and also highlights the importance that the perceived ease-of-use as on perceived usefulness. The perceived ease-of-use has a positive influence in the perception of enjoyment, showing that the easier it is to use, the more the application is enjoyable.



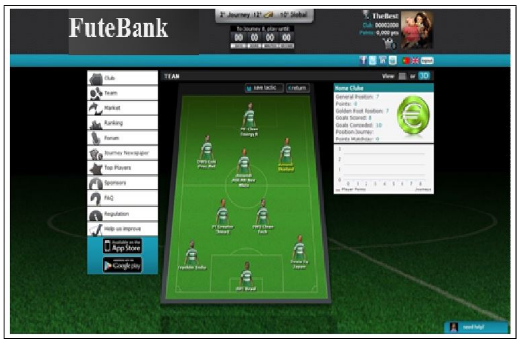
	value (on EUR)	no. of shares	remove
1 Threadneedle Investment Funds - Threadneedle UK Institutional Retail Net GBP Acc	1,00		☐
2 ACMBernstein American Growth Portfolio C USD EUR			☐

management company: Threadneedle Investments
category: Acções
current nav: 1,1923 GBP
nav date: 03-06-2013
min. amount-additional: 500,00
min. add. quant.: 0,00
init. minimum amount: 500,00
init. min. quantity: 0,00
sell. date:
regulating entity: Financial Services Authority - 25, The North Colonnade - Canada Wharf - London E14 5HS

Please read the Fund's Prospectus and Legal Information available in pdf.

Total on EUR : 1,00

(a) Example of a traditional mutual funds purchase web page



(b) Example of FuteBank's mutual funds portfolio

Figure 17: Gamified e-Banking Application

As positive as the results were, limitations were also present. The game was only available for existing customers and represented a real investment with a minimum of six investment funds on their portfolio, the results cannot be generalized to all customers of e-banking, or future customers with or without the type of financial product used in this game. While the theoretical basis for the various perceptions analyzed in this game is supported in this empirical study, the implementation of various types of 'social' technologies in different contexts of e-banking requires further study.

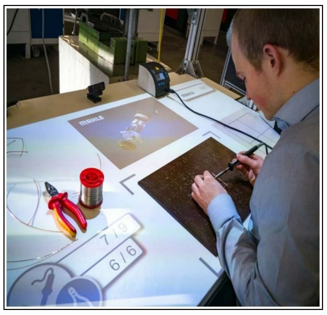
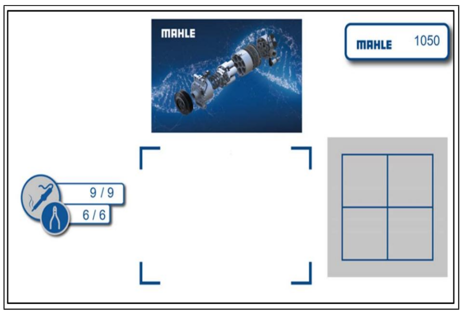
3.1.2 Workplace Gamification

Workplace Gamification can be considered a certain case of Gamification. Workplace Gamification integrates Gaming in the workplace to create a more interesting work environment. In our case for G4M, Workplace Guidance can be linked to Serious games, as the same guidance used in a Serious Game can also be used in Workplace Guidance, assisting workers while training and after training in a similar manner.

In [27] Funk et al. considered that giving workers guided assistance when working may not always be positive. When we are trying to assist someone who knows what he is doing, our guidance may not only be unnecessary but at some points even annoying or disorienting. As a result, guidance and workspace assistance should be adapted to the expertise of each user.

In their thesis, Funk et al. proposed calculating a user's expertise simply by calculating a percentage of how many of the user's moves were incorrect. As long as the user has a low error ratio, the system does not provide guidance, or provides minimal guidance. When the ratio exceeds some thresholds, the system provides extended guidance in three levels, expert, advanced, and beginner. As the proposed approach is very simple, the authors state it is crucial to design a more complex user model and adapt the provided guidance further in a more realistic environment.

Schulz et al. [28] performed an experiment testing if Branded Gamification improved productivity in a workspace. Branded Gamification is when asking workers to do a boring and repetitive task we present additional audiovisual information to brand the given task as a game.

(a) Participant soldering a grid of copper wire [28]

(b) An example of the workbench used in [28] study

Figure 18: Experiment Testing in a workspace.

Participants were asked to perform a soldering task, with the goal of producing two rectangles, in a square grid (see Figure 18b , bottom right section). This task was split into two distinct parts. First, participants were asked to cut copper wire into sections of similar length. This was followed by the soldering of the copper wire, to forming two rectangles, in a square grid. These tasks were deliberately set to be repetitive in nature. Prior to performing the task, the participants' personalities were measured through questionnaires, while during the exam the participants' emotions were continuously measured using Emotient FACET. The task was conducted on a worktable. During this process, an interactive system projected instructions and feedback into participants' workspace.

Overall, Schulz et al. found a dual nature of branded Gamification: While instigating positive emotions, aversion and reactance also arise. Besides, users experienced stronger fluctuations within the branded Gamification. A higher coefficient of variation was found in terms of the negative feelings, compared to the variation in generic conditions. Additionally, Schulz et al. found negative feedback caused spikes in negative emotions. Participants'anger time increased shortly after receiving an error message, or message indicating a task had taken longer than a minute to be performed.

3.1.3 Adaptive Training Environments

A virtual training environment has become mandatory for Machine shop training nowadays [6]. The main problem is there are multiple solutions, but their training success varies for every person trained. Loch et al. [6] also proposed an adaptive training system which adapts depending on how it evaluates its user, giving each trainee the optimal training for him/ her.

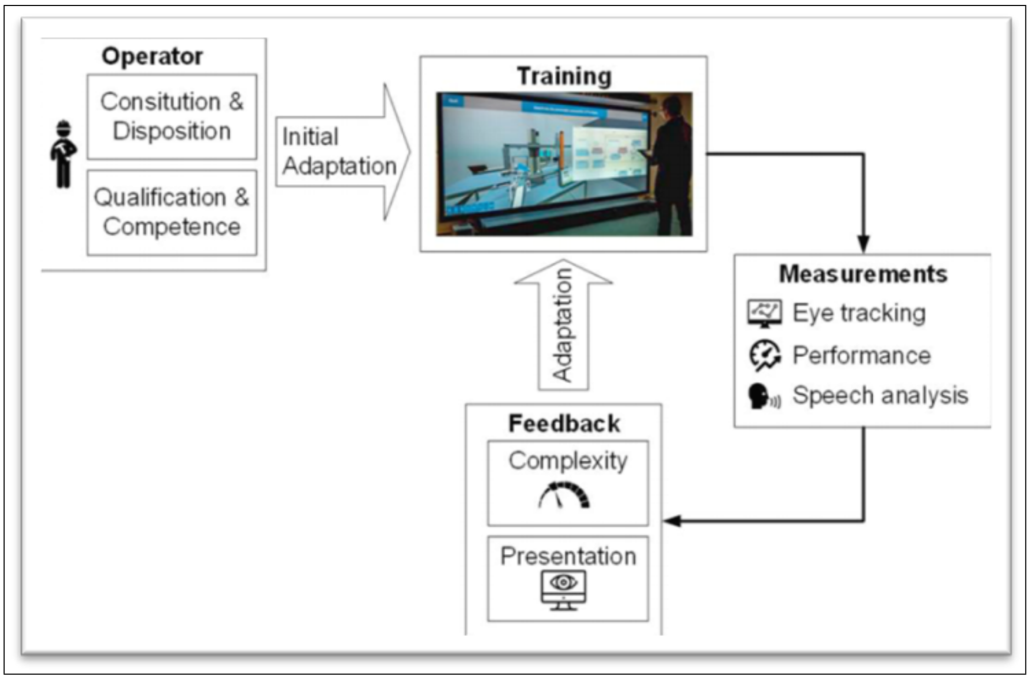


Figure 19: Adaptive system Architecture proposed by [6]

In order to adapt the training, a set of user characteristics is kept. These characteristics are divided into Constitutional (e.g. gender) which cannot be changed, Dispositional (e.g. personality) which change in a lifetime, Qualifications (e.g. knowledge) which is influenced by the individual's performance and Adaptation (e.g. fatigue) which depend on the situation.

In the proposed approach there are two steps to adapt the training process (Figure 19), an initial adaptation of the system and a realtime iterative adaptation. The first initialization is done by using or estimating the aforementioned user characteristics before the simulation is executed for the first time. Iterative adaptation is based on three types of measurements: perception, cognition and action.

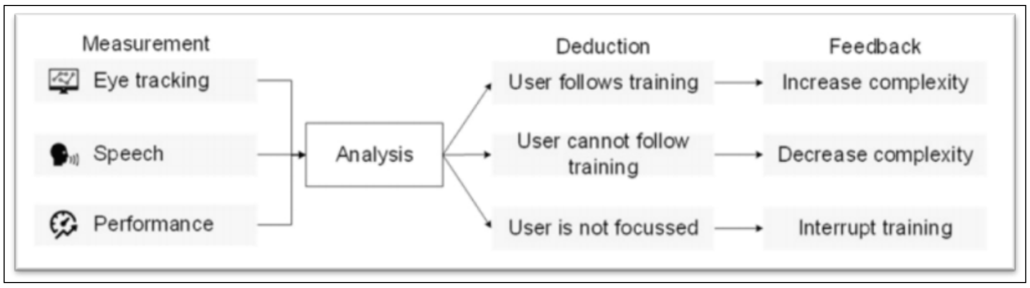


Figure 20: Measurement analysis system

Depending on the measurements taken during Iterative adaptation, the lessons may change in three aspects (Figure 20): interactivity, presentation and content. Interaction describes how the user communicates with the training system. Modifying this aspect allows providing the user with an input modality that fits her or his capabilities, for example, users with declines in motor capabilities can use a speech-based interface, while Simple interaction modalities (e.g., keyboard) can address users with low computer literacy. Presentation describes how the training system communicates with the user. The font size or the color scheme can be adapted to support users with decreased vision. Similar adaptations could be done for verbal presentation, for instance by changing the playback volume of spoken information output. Content describes which information the training system provides about a work step. Instructions can be combined to functional groups with increasing levels of abstraction. Information, for instance the necessary tools, can be removed from the instructions if the user is already aware of them.

This approach was based on their previous work [29] where Loch et al. developed a Virtual

Training System which was tailored specifically to training aging employees for machine training. Taking into consideration specific characteristics and weaknesses aging employees have, Loch et al. adapted a standard serious game specifically to the needs of aging employees. As they stated in their work, the average age of the active workforce in Machine Operation is rapidly increasing so training systems must adapt to the needs of training aging employees.

Because this training system is focused on aging employees, this training system also takes into account some specific requirements. When compared to younger trainees, aging trainees have more difficulty in performing complex tasks which require fast or accurate movements, short term memory or spatial perception in a complex or noisy environment. On the other hand, aging employees can use their long term memory and experience to perform tasks in a simpler manner without needing to take on-the fly decisions.

The proposed virtual system provides shorter lessons, which do not require much short- term memory, with textual and graphical descriptions of the work that needs to be done. Trainees see the virtual environment in a wall-mounted display or projector, while they use a tablet as a control panel for the machines. Graphics are simple without many colors or many buttons (Figure 21), so they are less prone to distract employees with lower attention span. The training process has three forms, step-through execution, slideshow and interactive simulation and trainees can view the same simulation using their preferred method.

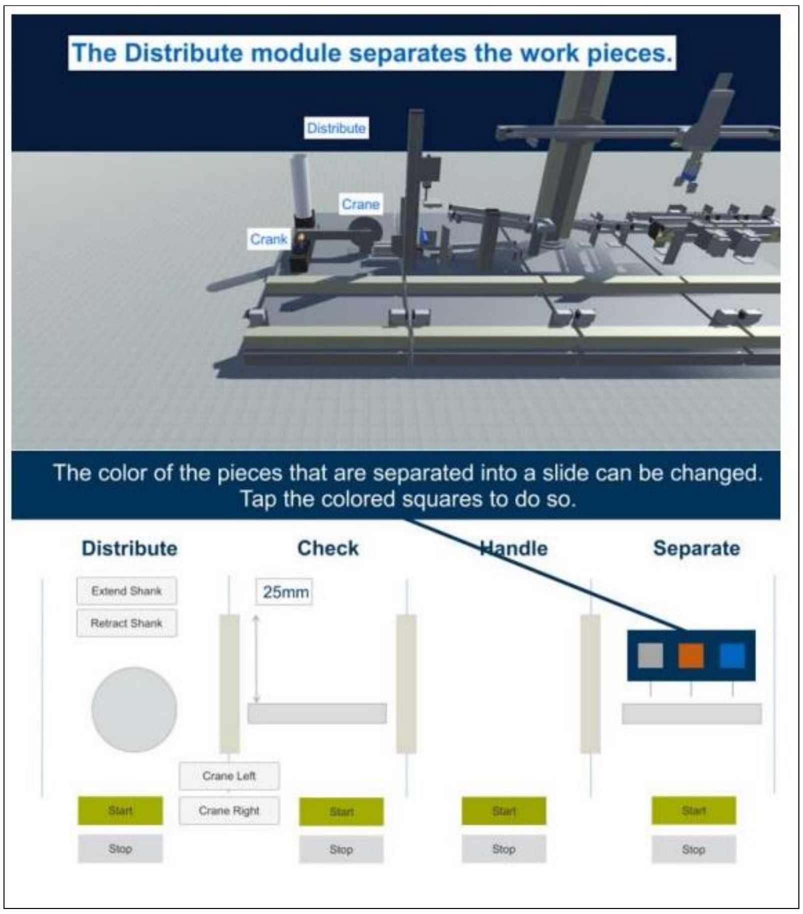

The Distribute module separates the work pieces.
Distribute
Crane
Crank
The color of the pieces that are separated into a slide can be changed. Tap the colored squares to do so.
Distribute
Check
Handle
Separate
Extend Shank
Retract Shank
25mm
Crane Left
Crane Right
Start
Stop
Start
Stop
Start
Stop
Start
Stop

Figure 21: The adapted representations of an instruction on the display (top) and the control panel (bottom).

3.1.4 Realtime prediction and guidance

Comprehensive mobile information provisioning plays a key role to facilitate flexibility and efficiency of operations in the smart factory [30]. Workers and production supervisors have to monitor the process state and its performance in real-time to react on incidents and communicate solutions immediately. For instance, if an input material in combination with a certain machine set-up in a manufacturing step usually leads to performance problems in subsequent steps, real-time prediction capabilities should warn workers during process execution and provide precise action recommendations on how to avoid the problem. This requires information about the current state of the entire process and its performance regarding metrics such as duration or quality. Additionally, information about work instructions and improvement suggestions, that is, process knowledge, as well as process communication has to be available. Besides, all this information has to be provided in a situation-aware manner to make adequate use of it anytime and anywhere on the factory shop floor.

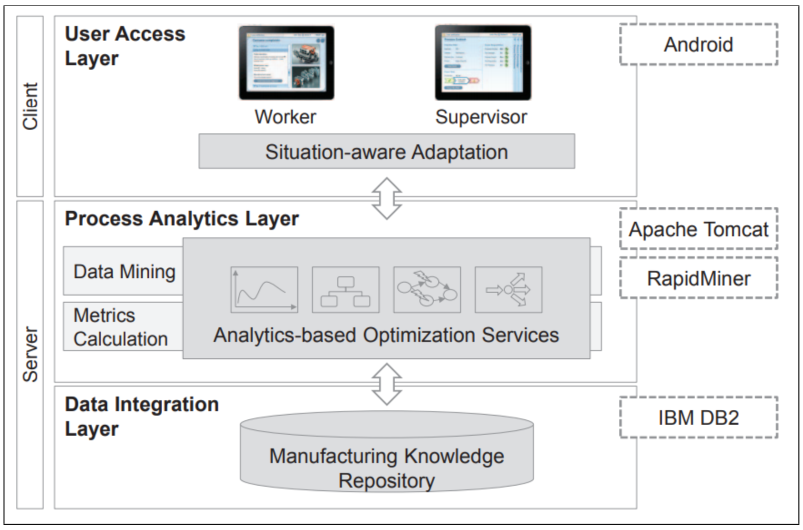


Figure 22: Architecture of the Mobile Manufacturing Dashboard [31]

In their paper Gröger et al. [31] present the Mobile Manufacturing Dashboard (MMD), a situation-aware manufacturing dashboard for mobile devices. The MMD provides advanced analytics and addresses the full range of process-oriented information needs of both shop floor workers and production supervisors. Thereby, the MMD is based on our Advanced Manufacturing Analytics framework and can be seen as an application of context-aware computing to manufacturing in order to provide situation-aware information on the factory shop floor.

Gröger et al. present the MMD as well as its underlying architecture. It comprises three integrated layers and makes use of their Advanced Manufacturing Analytics framework.

- Data Integration Layer comprises the Manufacturing Knowledge Repository
- Process Analytics Layer
- User Access Layer

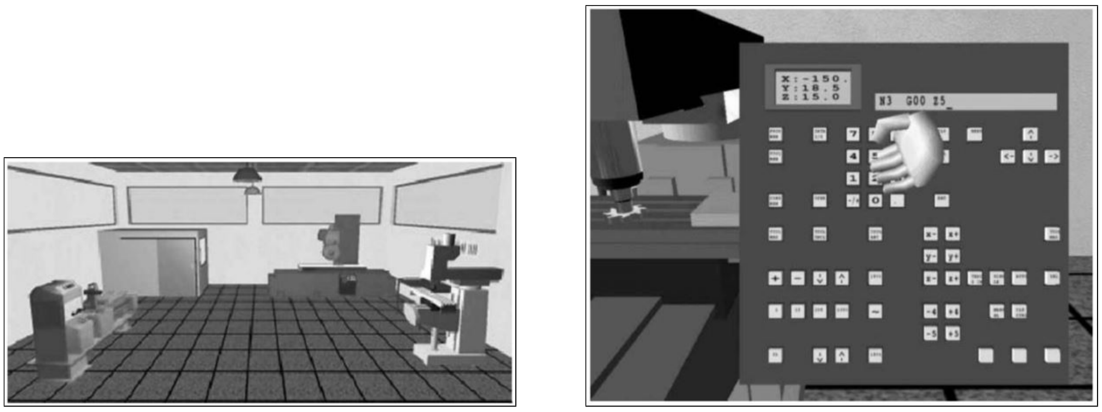
Finally, they described two representative real-world scenarios for a demonstration of the MMD, one focusing on a worker and the other focusing on a production supervisor.

3.2 Virtual Training Environments

Virtual environments refer to fully virtual games that attempt to recreate the real world. Their goal is to educate the users about the way real objects operate without bringing them in contact with the real objects at all. This is useful in cases where equipment is expensive and misuse while training may cause serious injuries or costly damages. It is also useful for educating a large number of trainees without using multiple real machines and large training spaces. The most common examples of fully Virtual Serious Games are Desktop Applications (using a screen, keyboard and mouse) or Virtual Reality Applications (using a VR Head-Mounted Display, optionally paired with controllers).

3.2.1 Virtual Machine Shops

Machine shop training is an expensive and dangerous process. Untrained users may damage or break expensive machines resulting in large costs or even serious injuries. This is why it has become mandatory to use virtual environments to train new users [6]. Because in G4M we will be developing a Virtual Machine shop ourselves we will first analyze previous work in the same topic.



(a) The VM Shop by Chryssolouris et al. [4] (b) NC Code programming interface

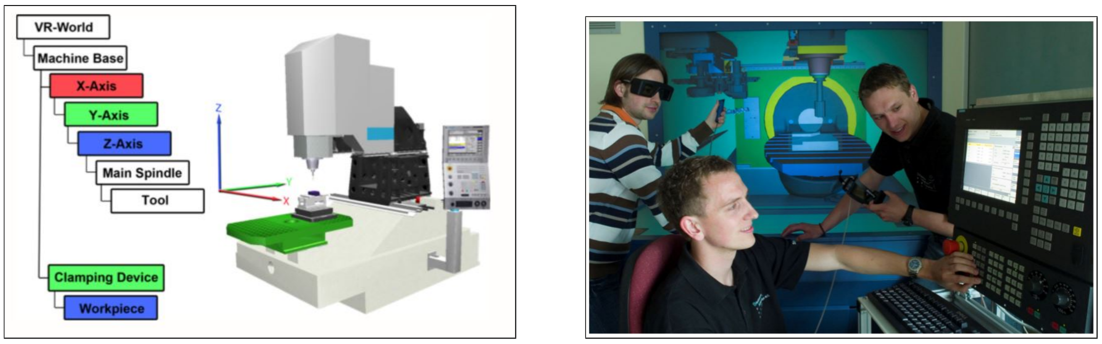
Figure 23: Virtual Machine Shop (VM)

The concept of a digital factory can be defined as an interactive 3D environment supporting modeling, communications and operation of manufacturing [5]. Chryssolouris et al. [4] designed a virtual machine shop in order to train new users in the usage of three different machines present in a machine shop. In this paper they clarify the reason for making their simulation is that a virtual simulation is considered mandatory for training new users as mishandling machinery result in damaging high cost equipment.

As a result, Chryssolouris et al. decided to design a virtual Machine Shop in a Virtual Reality environment to further immerse new users in the simulation. In the Virtual machine shop players could manually operate machines through virtual interfaces, identical to the real machines, or even write NC code which could also be imported directly to real world machines. Users could see their code being executed in the virtual environment by realistic simulation of the machine's movement.

A 3D object was deformed to show the object deformation realistically using quadtree-based partitioning and then comparing the produced 3D object with the optimal model. The engraving/milling tools also checked if they were used appropriately or if the real world's tool would break from misuse, in which case the user was warned and the process was terminated. To further improve the simulation, time and cost values were also taken into account to give a better estimation of how optimal the users actions were.

Currently, to learn how to operate a machine tool – either because it is a new technology or machine or because someone is undergoing vocational training – is a mix of theoretical lessons and hands-on training at the real machine. For the hands-on training a real machine with the right setup has to be available. And even if that is realizable, it is not (or only with great effort) possible to show every detail of the process or of the machine and to let every student practice as long as he or she needs.



(a) Hierarchy of the model of the DMP45 V linear 3-axis milling machine [9] (b) Real NC control unit controlling a virtual model of a milling machine

Figure 24: Hierarchy - Real NC control Example

In Purzel et al.s' [9] study proposed a combination of a real NC control unit and a virtual machine model (NC-VR coupling) for the main part of the training. A solution for this problem was the integration of the real NC control unit within the simulation environment. The coupling between a real control unit with a simulation environment is called Hardware-in-the-Loop (HiL) [32].

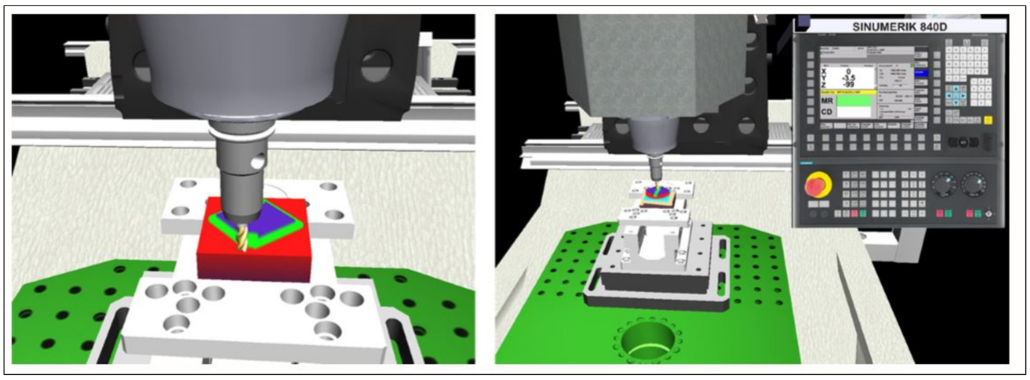


Figure 25: Virtual milling simulation at the 3-axis milling machine

Similar studies like Xiang Yang et al. [33] presented an approach to support analysis and design of manufacturing systems at different levels. The combination of various tools and the integrated use of them enable the study of a holistic application scenario. Changes to a manufacturing system can be applied virtually in order to minimize impacts on the real manufacturing system. This can improve the planning quality, accelerate the planning velocity and avoid production shutdowns. This paper consists of 4 sections as follows.

- The PVR framework.
- Three virtual factory tools
- Sound simulation and the planning of engineering changes (ECs).
- Two applications are visualized in a cave automatic virtual environment (CAVE).

PVR framework for manufacturing system design.: In this section, a Projected Virtual Reality (PVR) process framework is introduced which is divided into three main phases: modelling, application and adaption (see Figure26).

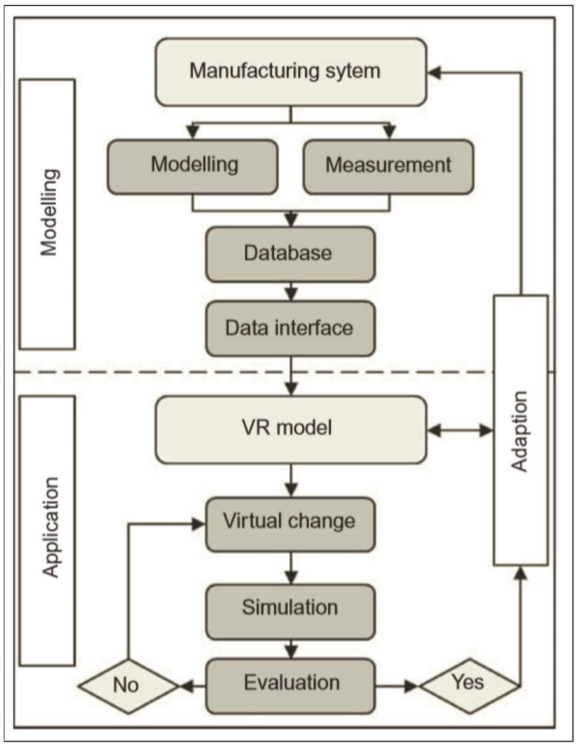
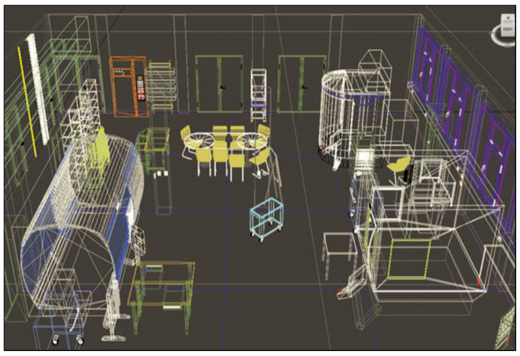


Figure 26: Workflow of PVR framework

Virtual factory tools.:

- **Modeling** - In the modelling phase, a data basis is prepared regarding the purpose of use. Usually it starts with geometric and mathematical modelling (Figure27a).
- **Application** - In the application phase, two main components should be included: simulation and visualization. Simulation is the basis of the whole work flow. Based on the modelled virtual world, the simulation rebuilds the manufacturing processes and provides essential information for visualization and PVR.
- **Adaptation** - Using the results from the application phase, the manufacturing system is adapted. A PVR-supported Continuous Improvement Process (CIP) workshop is used to implement such adaptations (Figure27b).


(a) Geometric Modeling

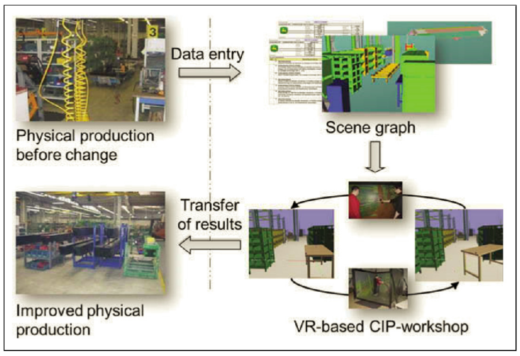
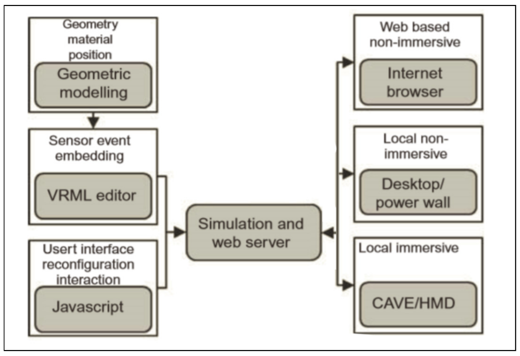
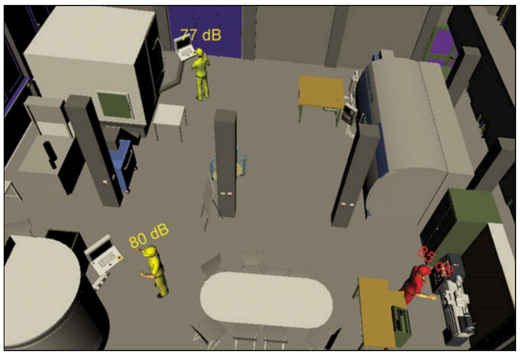

Physical production before change
Data entry
Scene graph
Transfer of results
Improved physical production
VR-based CIP-workshop
(b) Procedures of PVR-based CIP workshop

Figure 27: Virtual Factory Tools - Modeling and Adaptation

Sound Simulation investigates the noise issue in industry and uses a simulation and a PVR-supported method. The visualisation of simulation results and enhanced analysis capabilities in PVR provide a new point of view to understand the noise issue. Furthermore, this application can meet the requirements of noise control/reduction during factory planning.



(a) Structure of the sound simulation tool

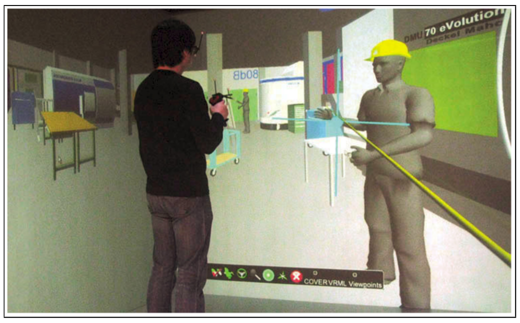


(b) Evaluation of sound levels

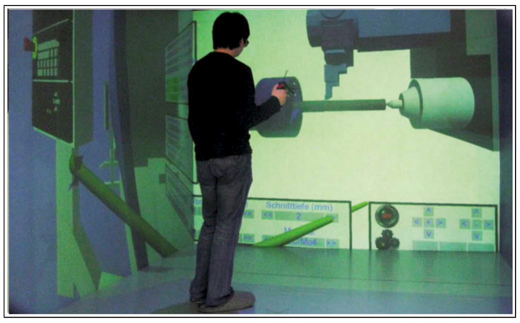
Figure 28: Sound Simulation

Application in a cave automatic virtual environment(CAVE): Figure 29a shows a user exploring the virtual factory in CAVE and simulating the sound to analyse the workstation or layout planning. Users are able to investigate the virtual environment much easier using a fly stick than using a mouse and keyboard in front of a desktop display. The virtual employees are scaled according to a real person, so that the users have a more realistic perception.

Figure 29b shows an animation of cutting process is visualized in the CAVE. One of the advantages of immersive animation is the free choice of viewpoints. Users can navigate to any relevant investigation point. Different aspects can be analysed regarding specific problems. Another advantage is that the full immersive environment with user tracking system improves the result of the virtual operation training.



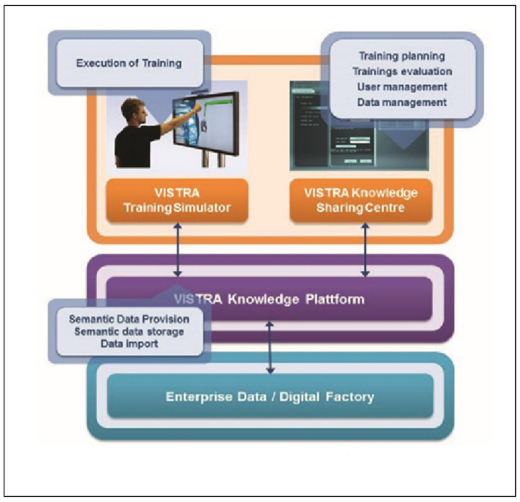
(a) Structure of the sound simulation tool



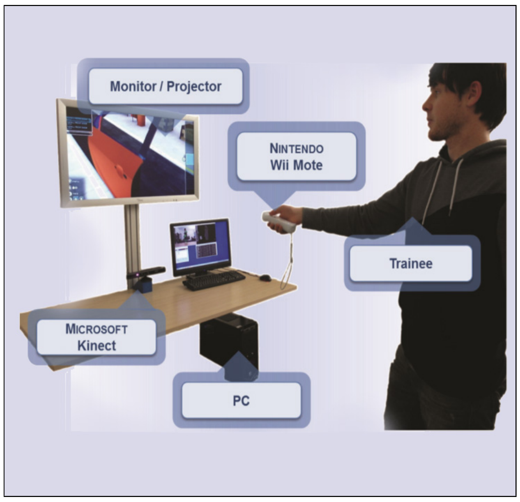
(b) Evaluation of sound levels

Figure 29: Application in a cave automatic virtual environment(CAVE)

The purpose of Nerstor Ordaz et al. study [34] was to determine the impact of gaming experience on the learning process of a manufacturing operation using the VIrtual Simulation and TRAining (VISTRA) system with architecture that shown in Figure 30a, a serious game that simulates manufacturing environments in order to train operators to perform manual tasks. The simulated operations take place at a welding workstation for truck chassis parts, where automation and manual tasks are combined. The case study aims then to evaluate the impact of gaming-experience and the general usability of the VISTRA system. Each operator that participated, completed five different training scenes on three difficulty levels each using appropriate hardware setup (Figure 30b). Completion time and mistake count were computed by the VISTRA system after completing each training scene. This information was analyzed and compared. Results showed that: (1) users without gaming experience took considerably more time to complete the sequences than users with gaming experience, (2) the same amount of mistakes were made by gamers and by non-gamers, and (3) 50% of the mistakes were made during a particular scene. The study thus found that gaming experience influences positively on training completion time using the VISTRA system. The particularities on the mistake count demonstrated that gaming experience does not influence the understanding of the manufacturing operation.



(a) VISTRA High-Level Architecture



(b) VISTRA Hardware Set-up

Figure 30: VIrtual Simulation and TRAining (VISTRA)

3.2.2 Virtual Assembly Simulations

With the recent advances in computer graphics, virtual reality (VR) is becoming a popular technology in many industrial applications that require a realistic computer–human interface. Traditionally, the graphic rendering of computer aided design (CAD) software focuses on mathematically accurate simulations. This accuracy takes priority over the rendering speed in CAD. In contrast, VR applications focus on a more intuitive and immediate understanding of simulated objects. For VR applications, real-time rendering, interactive and immersive visualization are more crucial, since VR attempts to replace the user's perception of the surrounding world by a computer-generated environment, which is often called a 3D virtual environment. Users can visualize and interact with the environment and can directly manipulate objects in this virtual world. To some extent, VR techniques have been successfully applied to scientific visualization, flight simulation, telepresence, training, education, medicine, entertainment and, most recently, to manufacturing.

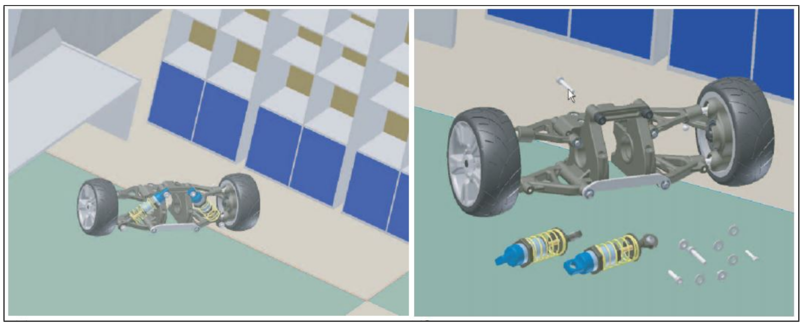


Figure 31: A vehicle suspension system with 119 assembly occurrences with related constraint-based manipulations in VR environments for disassembly process.

In their paper, Wang et al. [35] present another type of VR and CAD integration for virtual assembly, named a CAD-linked virtual assembly environment. It implements a virtual assembly application by external integration with a CAD system. More specifically, the VR application runs in a separate process but keeps linking and communicating with another process on the same computer, which runs the CAD application. In most cases, the CAD application runs in an invisible model. The data integration between VR application and CAD system is supported by a novel hybrid virtual assembly model maintained in the VR application. The hybrid model works to combine precisely a represented design dataset in CAD system and a hierarchically structured virtual environment dataset of low resolution in the virtual environment.

Towards the real application of the proposed virtual assembly environment in production, further investigations are also needed on how to support advanced assembly constraint types in this VR environment. Beside commonly used assembly constraint types in CAD systems, the advanced assembly constraint types referred to Wang et al. study [35] can be motion constraints or transitional constraints etc., which are getting supported by modern CAD systems.

Vosniakos et al. [36] propose an Application of immersive Virtual Reality that is advocated for assessing human-based assembly of large mechanical parts, including assembly jigs and fixtures as well as tools used and procedures followed. The primary aim of this Application is to enable subjective/customized assessment of the assembly system by the participating human. The secondary aim is to allow recording of the human's main movements in order to analyze them with standard ergonomics tools. Proof-of-concept was sought by a case study, i.e. aircraft wing assembly by riveting, involving a real human worker holding a real tool, whilst all the rest, i.e. avatar, wing, fixtures and factory were virtual. Health risk from assembly tasks was subjectively assessed via a questionnaire. Assembly equipment and procedure were preliminarily assessed by interpreting and applying RULA and REBA protocols. The potential to automate such a tedious task was demonstrated by automatic calculation of body postures from avatar's kinematics using special setup figure (33).

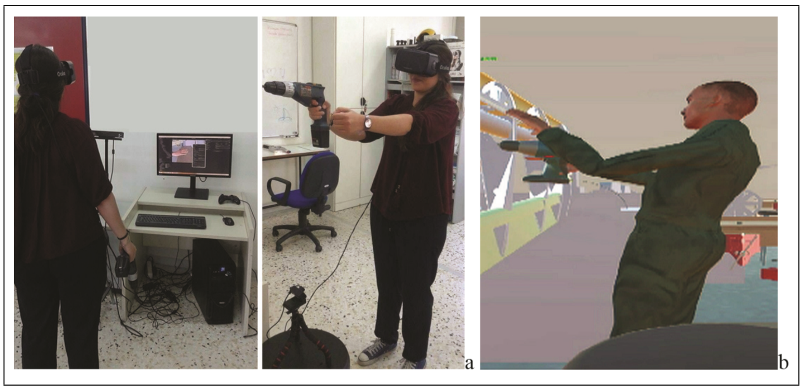


Figure 32: Riveting in the immersive virtual environment (a) human operator (b) corresponding avatar

Concluding, in assembly of large parts, such as aircraft wings, it became possible to try out the methods and tools prescribed in a low cost Virtual Environment. Although basic motions and interaction were successfully modelled, there is still room for future refinements for attaining higher fidelity in motion by further exploiting the game engine's functionality. The VR equipment is deemed adequate regarding accuracy, with some reservation concerning stability of viewing through the HMD and freedom of movement due to the sensor ranges and cable connection.

The avatar (mannequin) employed has enough joints to model basic but not subtle movements. Capabilities for finger tracking through wearable IMUs are currently being considered to rectify this discrepancy. All assessments performed so far concern reach and posture ergonomics as based on kinematics in 3D space. Haptic feedback of the equipment is an obvious future extension. Subjective assessment of the assembly setting by the participating human is advocated as equally important to objective assessment based on ergonomics protocols. Currently human operator motion is recorded on video and this is studied manually to detect patterns of undesired postures according to established rules in RULA and REBA protocols.

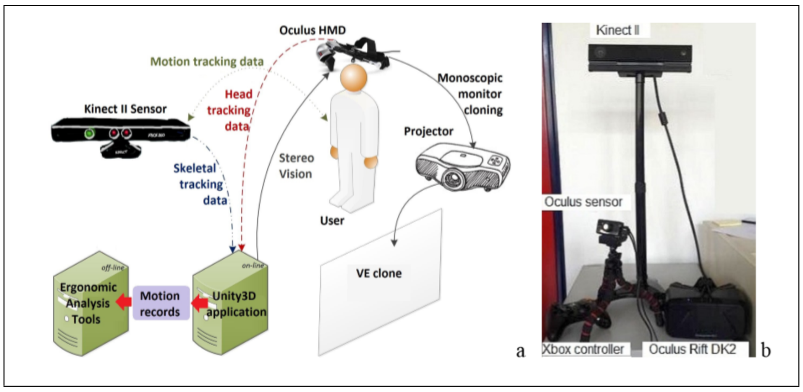
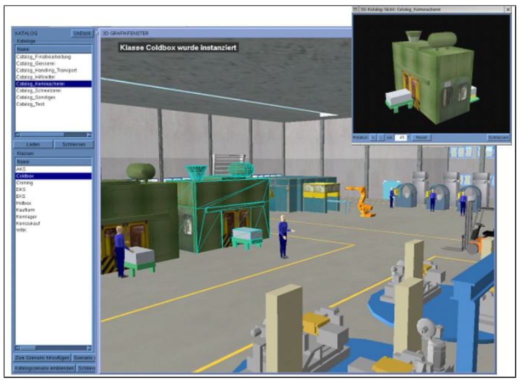


Figure 33: (a) Virtual Assembly Process Assessment for Large Parts; (b) Sensors employed

Another example is shown in Shcenk et al. [37]. The approach described in this article combines two approaches, i.e. assembly simulation and virtual reality, to create fully-interactive and immersive 3D visualisations of assembly lines and factories. In such virtual environments users can experience and interact with the virtual representation of their factory. Usage scenarios include the design phase of the factory as well as the operation phase. Targeted end users include classical factory planners, decision makers and even the individual assembly worker, who can train assembly tasks in a virtual environment (see figure 34).



(a) Easy model creation by using predefined library components

(b) Visualization of a cylinder head production

Figure 34: Automotive assembly technologies

The overall objective of the projects described in this article is to provide a virtual-interactive environment which supports the entire life cycle of a factory. This includes the usage of VR as a command and control tool for factory design and factory operation. The virtual world can act as the integration platform for different simulation models. In the design phase of the factory the processes of the planned factory can be tested and optimized within the VR world. This very world can be enhanced by didactic knowledge to qualify workers for assembly procedures.

In Michalos et al.'s study [38] shows that the automotive assembly systems are subject to many technological transformations because of their need to adjust to a continuously changing market. Manufacturers, in their effort to remain competitive, seek new technologies and equipment, which will allow their companies to increase their responsiveness to demand fluctuations and variability. Agile, modular and autonomous assembly systems are considered as the most suitable solutions. Every process in the assembly plant has to become more flexible in order for the overall system's flexibility to be increased. Therefore, new technologies are introduced to almost every section of the assembly plant. New applications have shown the potential benefits of these technologies as well as the specifications that future assembly plants will have to fulfill in order to respond successfully and cost effectively to shifting market demands. These specifications include:

- Flexibility and adaptability assessment capabilities in order to account for them in the decision making process and further improve the plant's responsiveness.

- Flexible equipment, in most areas of the assembly facilities, including body build, paint shop and final assembly.
- Synergy-collaboration between humans and robots in order for the benefits deriving from the human workers (decision making and intuition) to be combined with those from robots (speed, strength and accuracy).
- Fully digital planning and validation of assembly processes prior to any physical installations.
- Flexible and adaptable assembly technology and strategy, i.e. robotic fixtureless assembly, self-re-configurable assembly systems, increased modularity in the assembly process.
- Implementation of advanced joining technologies offering improved quality, productivity and safety
- Absolute customization should be the focus of all plants, following a pull/customer driven model instead of a push/ OEM driven model.
- Virtual manufacturing tools considering line balancing, but in a static way. There is a need for intelligent simulation tools to dynamically propose optimized mix and work distribution.

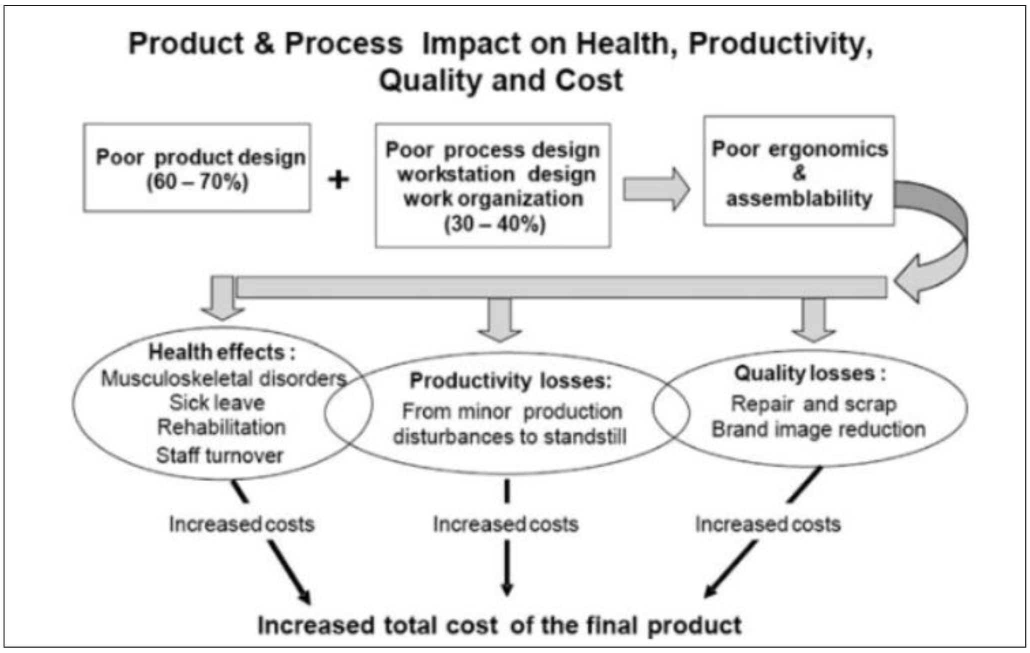

Product & Process Impact on Health, Productivity, Quality and Cost
Poor product design (60 – 70%)
+
Poor process design workstation design work organization (30 – 40%)
Poor ergonomics & assemblability
Health effects : Musculoskeletal disorders Sick leave Rehabilitation Staff turnover
Productivity losses: From minor production disturbances to standstill
Quality losses : Repair and scrap Brand image reduction
Increased costs
Increased costs
Increased costs
Increased total cost of the final product

Figure 35: Consequences of poor product and process premises seen from an ergonomic viewpoint Falck et al. [39].

There is also an Impact of Poor Assembly Ergonomics on Product 35. Falck et al. [39] confirms in her study a clear relationship between poor fulfillment of ergonomics requirements and quality errors in the quality assurance systems. A relationship between poor ergonomics assembly concepts and quality risk concepts was found in the manufacturing engineering phase of product and production development just before plant launch. In the assembly process and for factory-complete cars, it was found that there was a much higher amount of quality errors related to high and medium load level assembly items than to low load level items. The general conclusion is that there are huge savings and increased profit margins to make and great opportunities to increase the overall productivity and lower the selling prices of the final products by eliminating ergonomics risk solutions at early conceptual stages of the product development process.

3.3 Augmented Reality in workplace guidance

Augmented Reality is a unique approach when it comes to designing the virtual experience. Instead of creating a Virtual environment to cutoff users from the real world, AR is used to augment the users' experience in the real world. As such, it cannot be used similar to Virtual Serious Games. Instead, AR-based Serious Games enhance traditional learning experiences, such as reading books or being inside a real assembly factory, while providing virtual guidance as if the user would be guided inside a virtual environment.

3.4 Augmented Books

Ivanova et al. [40] used AR in conjunction with a machining guidebook to provide some 3D visualisation of what is presented in the guidebook.

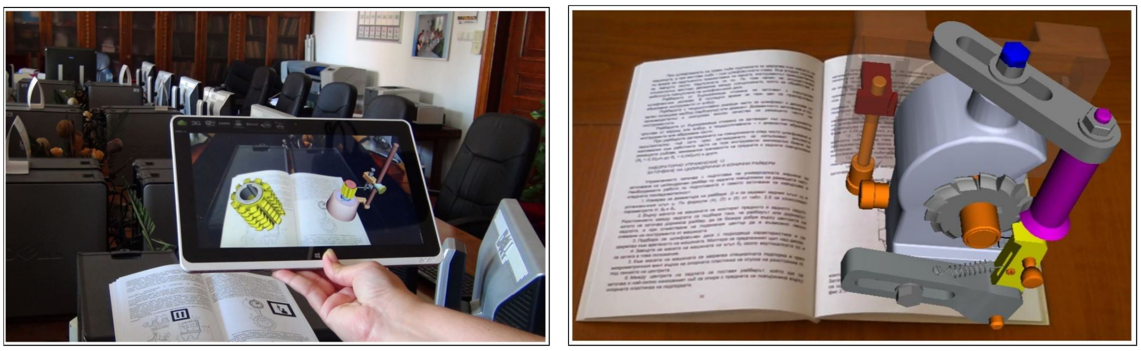

(a) AR view Using a Tablet (b) Superimposed AR Gear cutter

Figure 36: Augmented Reality in [40]

The guidebook used black and white square markers on the pages which were identified using a tablet and depending on each marker the correct visual content was presented to the reader in AR. Because machining objects are usually complex, it is difficult to present it in 2D drawings on a book, thus using AR the readers could see the 3D objects while reading about them in the book. Some of these AR objects were also animated to show how a machine works. Other objects were also interactable. By touching a complex machine for example it would disassemble into smaller parts and then reassemble to show how it is built. Finally, another use of AR content was to show live video tutorials of what is described on the opposite page (Figure 37) so users could see a live demonstration of the described tools.

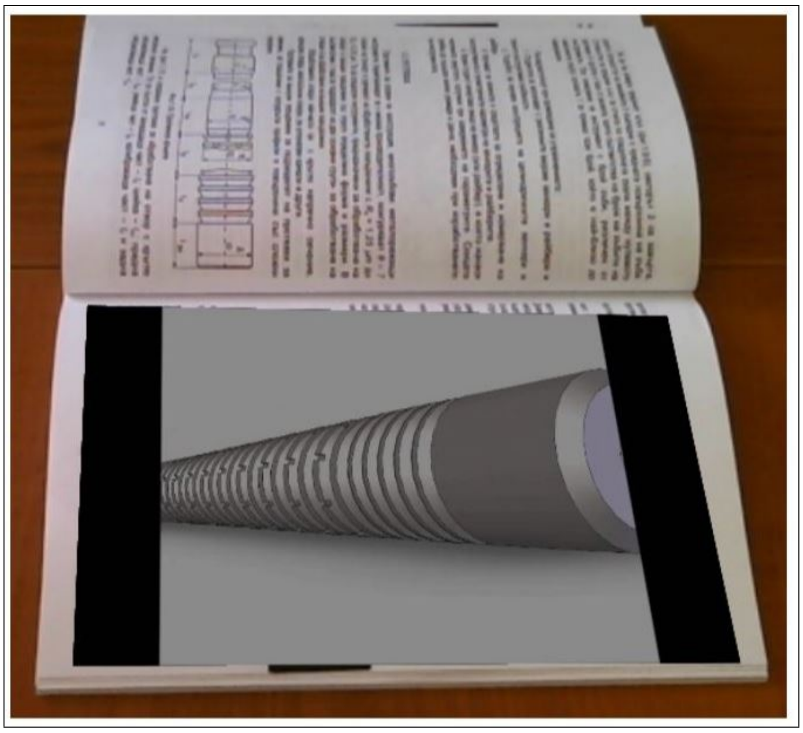
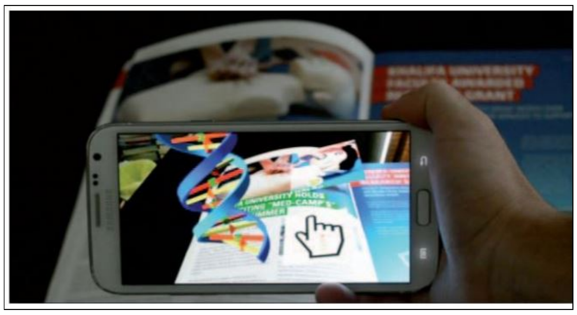


Figure 37: Video tutorial next to written guidance

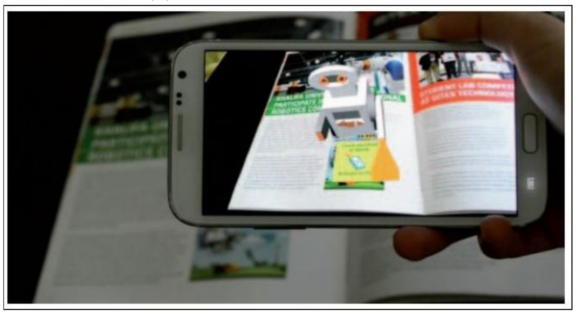
On another case, Bazzaza et al. developed iARBook [41], which was created to add immersion to already existing AR content. The basic system used Vuforia to track specific pages in a magazine and Unity to present AR content when Vuforia detected specific pages through a smartphone's camera. Bazzaza et al. then tested various methods of interacting with the presented AR content. The demo applications presented in their work consisted of: 1) A superimposed video over an article about a speech (Figure 38a), 2) a 3D model starts spinning and playing sounds with a touch on the touchscreen (Figure 38b), 3) "touching" a superimposed virtual tower to topple it (Figure 38c), 4) "pressing" a hyperlink on an article by hiding a text or button with the user's finger(Figure 38d, 38f) and 5) Using speech recognition to make a virtual robot speak (Figure 38e)

(a) Superimposed Video


(b) Rotating DNA Helix

(c) Pushable tower of Piza

(d) Superimposed flag


(e) Talking robot

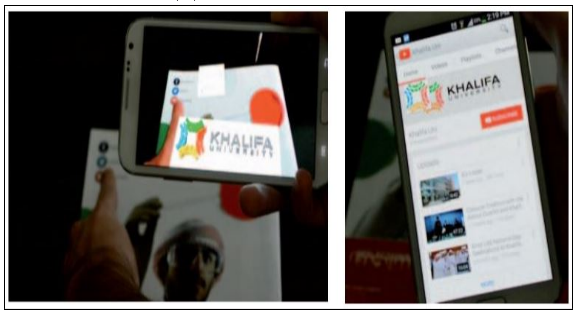

(f) AR Hyperlinks

Figure 38: Augmented Reality demos in [41]

At the end of the paper, Bazzaza et al. also presented an alternative that uses an immersive desk that scans the iARBook through a top-down camera instead of having to hold a smartphone. Using an immersive desk provides better quality in the IAR content presented and eliminates having to hold a smartphone while reading, but is a more expensive and non-portable solution.

Another approach in Serious Gaming in previous works showed that using Augmented Reality positively affected the educational and development processes by engaging young trainees while integrating the latest display technology with traditional forms of learning, training and manufacturing [42]. Studies related to Augmented Chemical Reactions that is a chemistry teaching application which facilitates the understanding of molecules behaviour and structure using a 3D user interface offering direct manipulation [43]. Students are able to get acquainted with different elements and learn about chemical reactions. Anther one study was AR platform for electromagnetic education is developed, offering motivation and engagement for learning. AR training was deemed essential for smooth lecture flow [44]. AR educational apps often focus on 3D visualization of complex objects. For instance, a mobile AR application was developed for undergraduate anatomical education [45]. The application can track plastic heart models located in the classroom, without the use of observable markers. After recognition of the real object, text labels are depicted above specific anatomical parts of the organ. Using virtual representations of hearts, students are able to self-learn outside the classroom. A similar approach [46] teaches anatomy education by augmenting 3D models of organs on the users own body. Since AR's beginning, combining hard copy material with AR showcased AR's positive effect in education [47]. AR applications combine superimposed AR models with touch, speech and audiovisual content. Overall feedback suggests that a book is much more likely to be read when combined with an AR application [48]. AR technology has also been employed in the education and training of mechanical engineers. A recent AR approach superimposed 3D models of cutting, measuring tools and special equipment on a text book [49].

In summary, previous research focused on superimposing AR elements onto hard copy text books, offering, though, limited interaction and off-line availability of complex 3D models for training. This work goes beyond plain visualization of AR content, putting forward a mobile graphical user interface facilitating the importing of projected 3D models for further geometrical transformations.

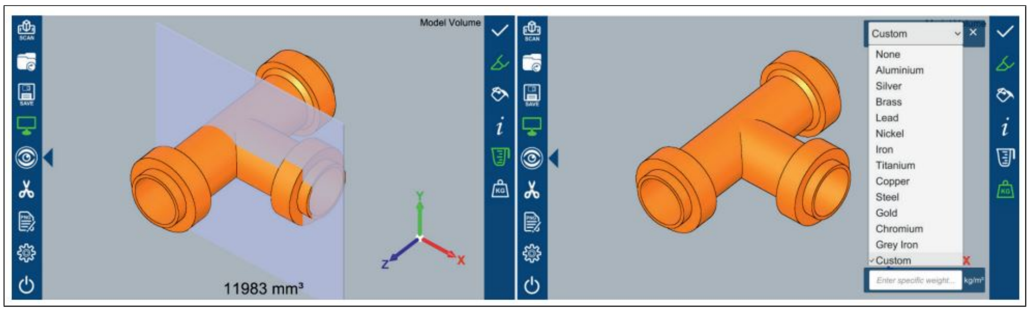
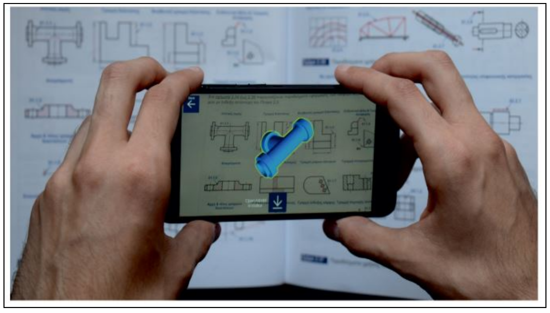
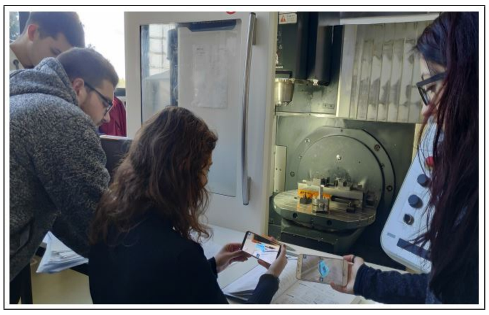


Figure 39: Volume (Left) and weight (Right) calculation. The result steadily increases until the whole model is scanned.

In this study Marinakis et al. [50] propose a mobile Augmented Reality (AR) application employed to facilitate and enhance the learning processes and textbook material of the Mechanical Drawing course for undergraduate and postgraduate university students undertaking an engineering degree. The presented AR app is used along with a brand new Mechanical Drawing textbook authored for the respective course (Figure 40a).



(a) Selection of a 3D model through Augmented Reality based on CAD model drawings printed on text book hard copy.



(b) Group of students studying mechanical drawing gathered around a CNC Milling Machine.

Figure 40: Mobile Augmented Reality (AR)application

3.5 Holographic AR Guidance

The use of Augmented Reality technologies, in support of the human operator, is becoming relevant to manufacturing. The evolution of the AR technology has proven to be an efficient key for problems, related to simulation, assistance and guidance for manufacturing processes [51]. AR is a novel human–computer interactions tool that overlays computer-generated information in the real world environment.

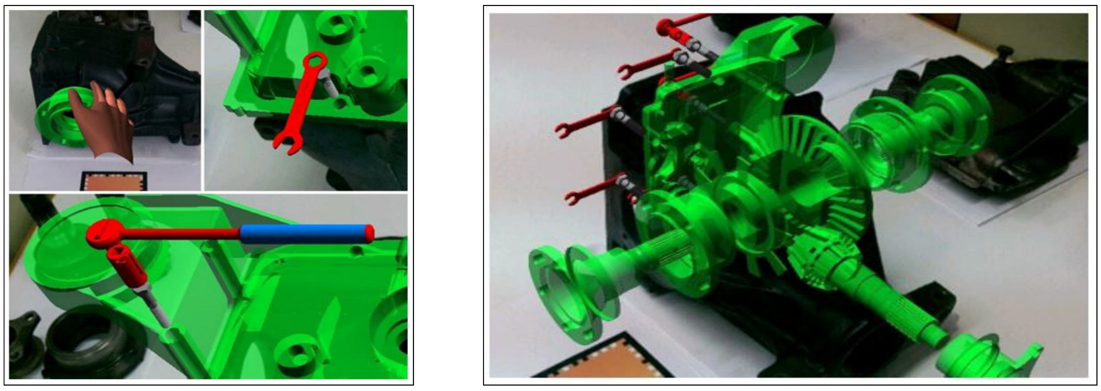
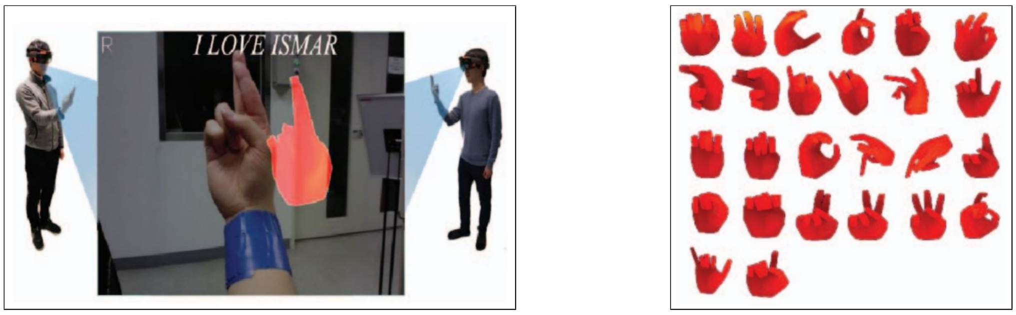


Figure 41: Augmented Reality for Human-based Assembly: Using Product and Process Semantics [52]

Rentzos et al. [52] present a semantic-based approach to building instructions for human-based assembly tasks to be used by Augmented Reality systems. This study has shown the way that Augmented Reality (AR) can be used as a final assembly guidance medium in assistance of the shop-floor operators, through the use of virtual assembly instructions. The techniques developed are provided as tools in support of the complex tasks, related to the assembly and are tested in a realistic test-case. The test case highlights the potential of the techniques and the usability and flexibility they provide to an engineer.

3.6 Human-Computer Interaction in AR

Although presenting AR content to users is a challenging task, it is also important for AR objects to be interactable. Most Mobile applications utilize the touchscreen for interactions as a simpler solution. There are also other more immersive solutions using Machine Vision or Voice recognition.



(a) Gesture recognition (b) Gesture dataset

Figure 42: The Gesture recognition proposed in [53]

Park et al. [53] propose a new method of hand-tracking for gesture detection. Tracking is done from a first-person view of a head-mounted camera (Figure 42a), which attempts to track the user's hand and recognise pre-defined gestures while the hand is in motion without the use of any markers. The gestures they aim to track are the letters of the American Sign Language (ASL) (Figure 42b).The ASL is composed of 26 different hand poses which alternate in quick succession, so a system that accurately tracks the ASL can be effectively used in multiple complex scenarios.

The solution proposed by Park et al. combines two tracking methods, a Convonutional Neural Network (CNN) and model-based tracking, as shown in Figure 43. Before sending any data to these methods the input image is preprocessed to segment the hand from the background, while a blue wristband was used to cutout the hand from the wrist. From the segmented image, a bounding box was created around the hand, which is sent to the CNN for classification. The Model-based tracker uses data from a depth camera at the same position of the bounding box. The output of the CNN is used as a second input to the Model-based tracker which selects the best matching option based on 26 hand features for each of its inputs.

The evaluation of the proposed method showed good results, even in fast moving hand gestures, except movements where the hand rotated around itself, where the position of the fingers and the depth of the hand remained unchanged.

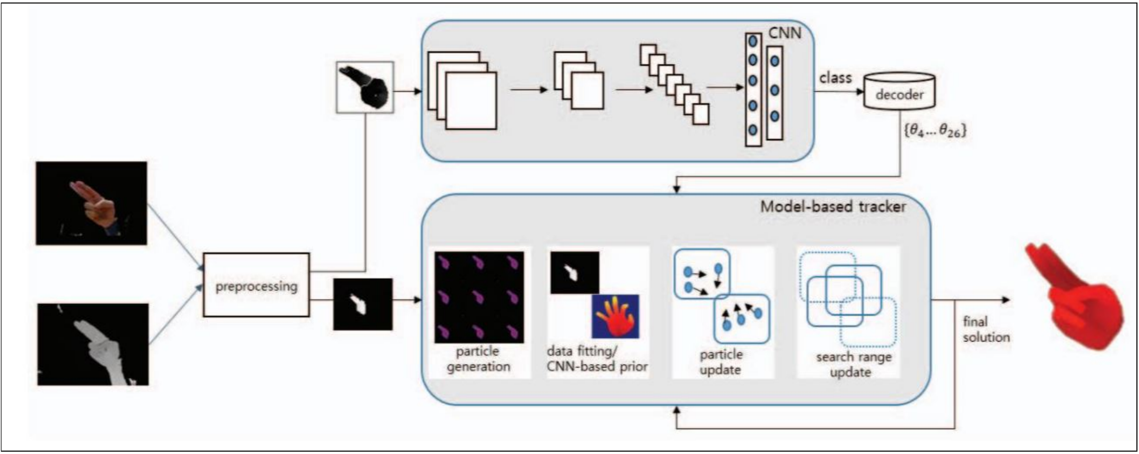


Figure 43: Park et al.'s system architecture

Other work such as [54] combine marker-based tracking and markerless tracking. The proposed system scans the environment using markers, while tracking the user's movements and gestures using a markerless Machine Vision based approach.

The proposed system provides both written instructions in AR and 3D guidance and the user is free to use any of these two types of guidance at will. The user also interacts with AR content using simple mid-air intuitive gestures. Results showed mixed results between written AR or full 3D guidance, but the proposed system outperforms standard digital documentation in a non-AR environment.

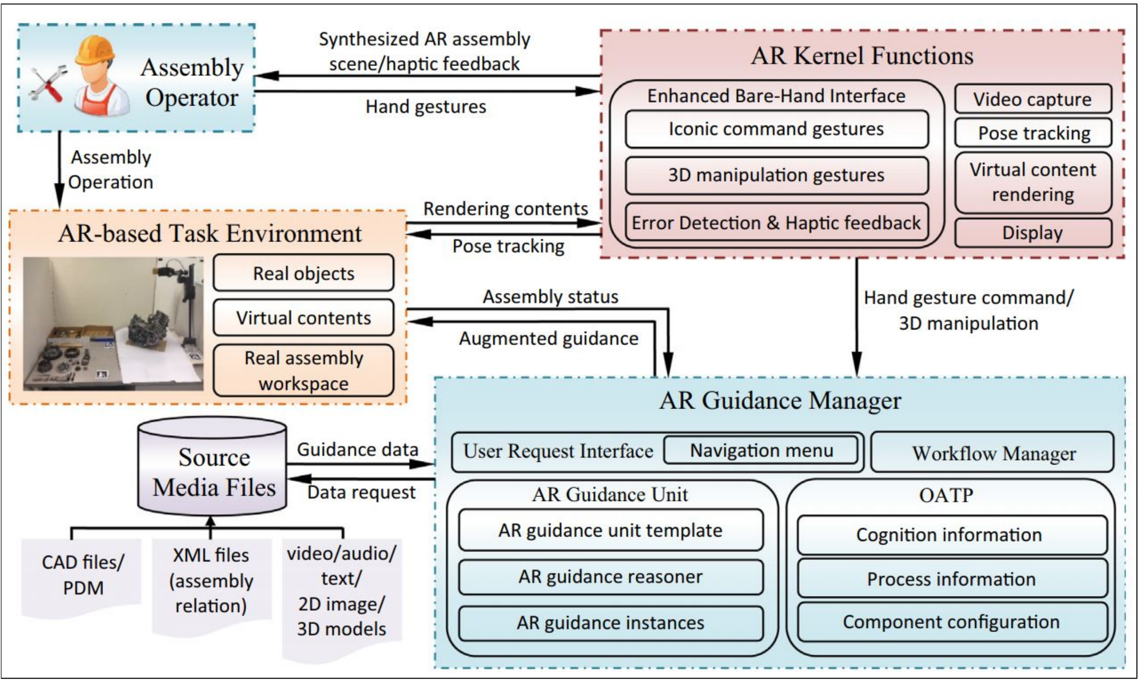


Figure 44: Wang et al.'s system architecture

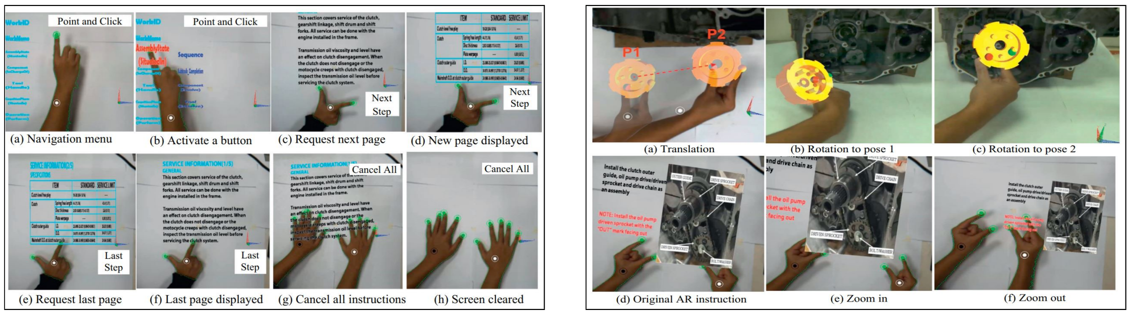
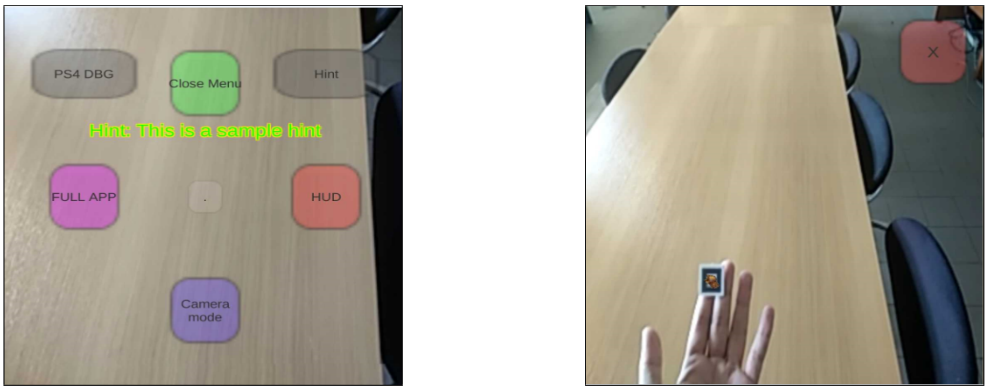


Figure 45: Gesture recognition used in [54] for 2D elements (Left) and 3D elements (Right)

Results show that the proposed system outperforms previous work on the same tasks, while both AR guidance systems vastly outperform standard digital documentation in a non-AR environment.



(a) Main Menu

(b) Cursor in FS-Mode

Figure 46: AR interface used in Holo-Board [55]

Holo-Board [55] presents a distributed system architecture (Figure 47) for use in AR Applications. This architecture aims to separate Machine Vision Applications from graphics applications and link them through a common channel. In the demo application of Holo-Board ARToolkit is utilised to track a 2D square marker which is placed on a ring worn by the user (Figure 46b). By tracking this marker the user can move a cursor and interact with superimposed AR elements (Figure 46a) in a similar manner to a computer.

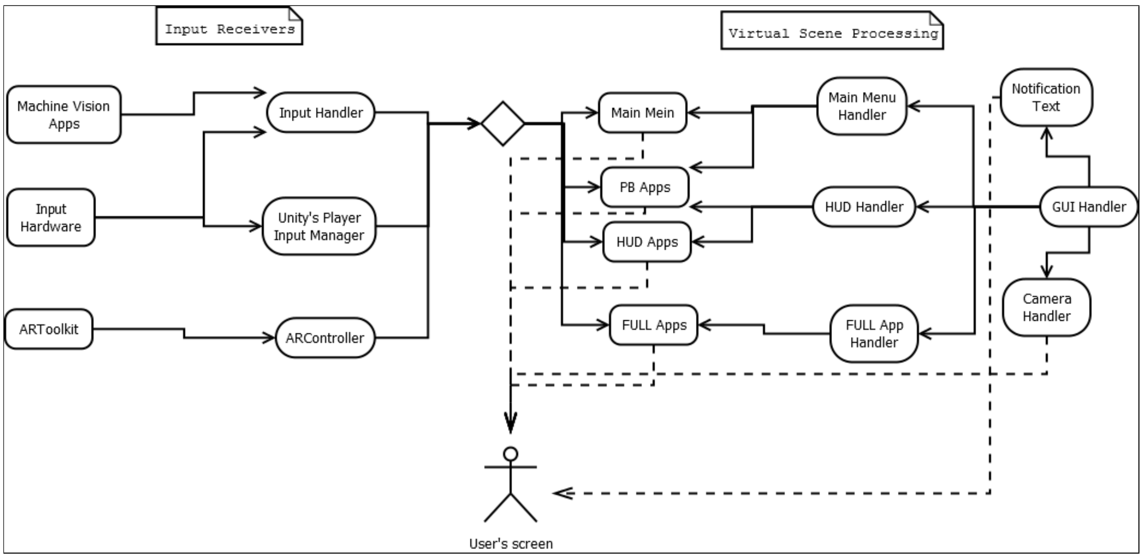


Figure 47: Holo-Board's system Architecture [55]

Holo-Board also states that a big problem mobile AR applications have is the lack of ways to interact with the user. Cardboard AR apps are a useful cheaper solution compared to AR HMDs, which cost thousands of dollars and are still in prototyping stages. By using the proposed architecture a Machine Vision programmer can create a generic tracking program which streams tracking info in a predefined format. Graphics developers can then use any Machine Vision algorithm with minimal to no changes in their code.

3.7 Interactions in AR versus VR

As both VR and AR are widely used in education and hands-on training for various cases, it is important to compare the usability of the two technologies. Kirakosian et al. [1] compared the use of VR and Ar in a dance training exercise for Salsa.

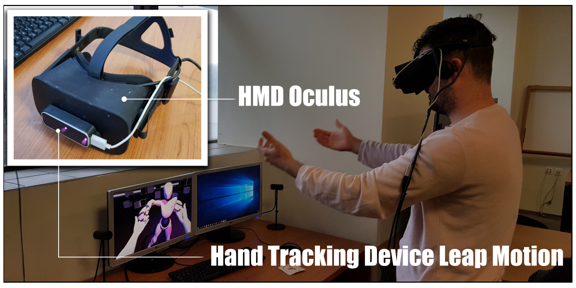
The VR application used a virtual avatar who could both show the user the correct steps of the dance but also dance alongside the user. The virtual avatar can both dance on his own as well as respond to the user's movements as a follower dancer by responding to the motions of the user's virtual hands and blending the pre-built dance motions with Inverse Kinematics (IKs) to make a realistic visual of the virtual avatar holding the user's hands while dancing.

The virtual avatar was modeled after a real-world Salsa dancer named Alberto Rodrigues who was scanned using an **Artec Eva White scanner** device. A **Microsoft Kinect** was used for Motion capturing of a real dancer's movements while dancing. The motion of the real dancer was imported into Motion Builder, where the captured motions were optimized and rigged to a virtual skeleton and extracted as avatar animations.

The Dance training app was designed as a VR and a separate AR application made in Unity Game Engine. The use of the Unity Game engine allowed the easier integration of the Oculus Rift SDK and Magic Leap SDK into the project while re-using the same assets and execution code with minor changes to the input handling.

In VR, the user could see their own hands inside the virtual scene with the use of a **Leap Motion** tracking device mounted on the HMD itself (Fig. 48a). The Leap motion was used instead of a traditional controller in order to give the user a stronger sense of immersion and freedom by leaving the user's hands-free. In order to make the two technologies more comparable a real-world environment was pre-selected and then the virtual scene was designed in the VR application to resemble the room of the real world (a room with white walls and wooden floor).

The AR Application used Magic Leap's SDK for hand tracking of the user's palm as well as Magic Leap's gesture recognition database to recreate Leap Motion's interactions, for example, the "pinch motion" is the transition between two distinct gestures: the "C" hand pose and "Pinch" hand pose. The hand position switched from the hand positions given by Leap Motion to one of the 15 distinct hand positions tracked by Magic leap, specifically the "palm center" position of each hand. The virtual scene and artificial lighting of the virtual scene were completely removed and were replaced by an invisible mesh geometry extracted from Magic Leap's spatial scanning. In AR, no virtual environment was designed, instead the scanned mesh of the environment was captured through the Magic Leap One SDK to detect the floor and walls and the virtual avatar was placed inside the real environment (Fig. 48b). The virtual avatar was able to interact with the environment through collisions with the scanned world geometry while being able to follow the user through Unity's Navigation Mesh AI system to walk realistically on any surface.



(a) VR setup

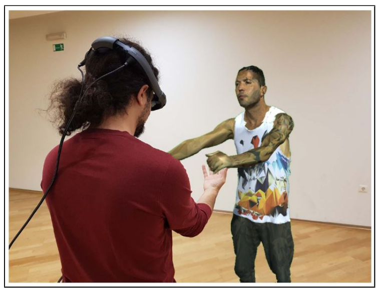

(b) AR setup

Figure 48: Experiment setups in VR and AR [1]

The training process for the user consisted of three steps. Initially, the user was shown the basic controls of the virtual environment, where the user could use various gestures to interact with virtual interfaces, such as "pinching" a button would initiate a "click", or that the volume slider could be adjusted by touching and moving the indicator on the slider. Next, the user was shown the Basic steps of the dance where the virtual dancer showed the dance moves one by one and the user was prompted to follow along. Finally, when the user felt confident they were prompted to dance with the virtual dancer. The user was then prompted to "grab" the virtual avatar's hands and start dancing. The virtual avatar followed depending on how the user dances by following the user's "leading" through the user's hand positions. The virtual avatar kept holding onto the user's hands while performing the dance animations through blending of Forward Kinematics (FK), which are the pre-made dance animations along with an Inverse Kinematic (IK) animation that keeps the dancer's hands attached to the user's hands without making the avatar's moves look unnatural.

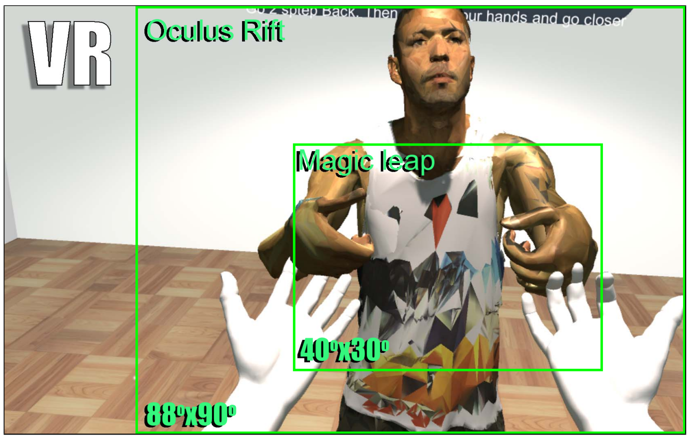


Figure 49: Comparison between the Oculus rift and Magic Leap One FoV [1]

The results of the experiments showed four key remarks:

- *(1) In terms of task load index, users felt they trained more successfully and with less effort in VR.* Through verbal feedback this is believed to be due to the limited AR FoV (Fig. 49).
- *(2) In terms of usability ratings, users expressed that the VR dance training system was more accurate and responsive as well as more fun.* Potentially, the complete immersion of the VR training system, excluding real-world distractions played a role in evoking higher ratings of perceived fun that in AR.
- *(3) Efficient navigation in VR was problematic.* Although users were aware of the wireframe that indicated the edges of the safe play area, users were still hesitant in navigating the play area due to fear of falling or being tangled by the Oculus Rift's wires.
- *(4) High variance of responses for the AR dance training system was present but not for VR.* This was mostly a result of personal preference. Certain users were more hesitant to use the VR display and thus were really welcoming to the less restricting AR HMD while others, especially those proficient in VR, found the graphical quality and limited FoV of the AR display extremely limiting for long term use.

In summary, while theoretically AR setup allows for more freedom of movement with faster and more accurate gesture recognition, interacting with large objects near the user, such as a human-sized avatar is less than ideal. Thus, AR objects should be scaled accordingly to accommodate for the limited FoV when using AR. In order for an interface to be clearly visible without covering a large part of the available FoV a general heuristic rule is approximately the size of the palm of a hand in 1m distance away from the user.

3.8 AR interface types

AR at its core has various forms. Mobile AR uses the smartphone as a handheld device the user carries with him, while HMDs, like Microsoft HoloLens, use a full first person perspective around the user. Other more dedicated systems present AR content on screens or projections on top of the real world, requiring a more dedicated setup, while not requiring the user to wear or carry any devices.

There have been many studies that have compared using HUDs to HDDs, for example in driving tasks. Doshi et al. [56] conducted a study with the windshield as a display and using AR elements to convey an alert when the driver drove above the speed limit. They found that user's had a faster reaction time using the HUD than when using a HDD. Similarly Liu and Wen [57] found that truck drivers were able to control their speed better when shown speed information in a HUD rather than a HDD, with a 1.0 second saving in reaction time. Medenica et al. [58] used a driving simulator to measure performance with an AR HUD compared to two types of HDD navigation aids. Using eye-tracking technology they found that with the AR HUD drivers were able to keep their eyes on the road significantly longer than with a HDD and experienced significantly less workload. Other experiments include the work of Charissis et al. [59] that found an AR HUD produced significantly less collisions in low visibility conditions than a HDD, and Kim and Dey [60] showing fewer navigation and driving errors with a novel AR HUD compared to standard HDD.

Some research has also been conducted on comparing AR to non-AR interfaces for the same display type. For example, Park et al. [61] compared driver performance when using world-registered arrows to screen-registered 2D icons on the HUD. They found that drivers had a significantly faster response time to lane changing information when shown with the world-registered cues. Similarly, Sawano and Okado [62] conducted a study comparing an AR HDD to a more traditional HDD with non-AR map interface. Users found the AR HDD interface to be more understandable, although there was no significant difference in driver performance.

A newer study by Jose et al. [63] (see Fig. 50) compared three types of AR displays Heads-Up Displays (HUD), Heads-Down Displays (HDD) and Head-Mounted Displays (HMD)

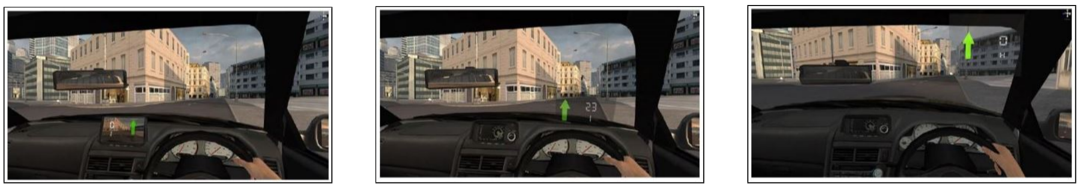

(a) HDD (b) HUD (c) HMD

Figure 50: The three Simulated AR views compared in [63]

The goal of this paper was to compare the three types of AR displays inside a driving simulation. Specifically, Jose et al. presented information such as driving directions, vehicle speed and if the driver exceeded the speed limit. Using a simulation they asked users to drive through specific routes while giving them all necessary route information through each of these displays. The simulation would measure their performance with factors such as errors made while driving, time spent over the speed limit as well as reaction time to sudden changes.

Overall, the results indicate HUD as the best in most area, yet each display having their own strength and weak points. In terms of navigational errors, participants performed better while using HUD compared to the other two displays. It is also interesting to note that there was a statistically significant reduction in the number of errors when using the HDD over the HMD, despite the HDD requiring the user to look down and away from the road. Despite being statistically better than the HMD at avoiding navigational errors, the HDD falls short in both the number of times it exceeded the speed limit and the time spent above the speed limit. In order to test how much the display forces the user to take his or her eyes off the road, virtual characters were added at places along the map. The HUD clearly wins over the HDD in this case, since with the HDD the user is busy with his or her eyes off the road and does not notice the models in the surroundings.

It is also worth noting for the HMD, the navigational instructions were not provided based on relative position of the head orientation, making it slightly more difficult to understand them. The size of the display areas for each AR Display condition was controlled to be the same, which could impact the results for the HUD and HMD.

When considering everyday interactions that use computer-aided systems it is important to define the purpose of each use and model them into categories based on their intent [15]. One such classification separates computer-aided technologies based on the intent of the user during the interaction.

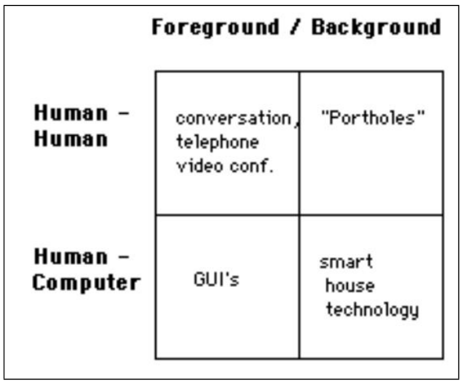


Figure 51: The basic communication model for computer-aided technologies [15]

In the proposed model (see Fig. 51), the interactions are separated into interactions based on the "ground" of the interaction as well as the expected target.

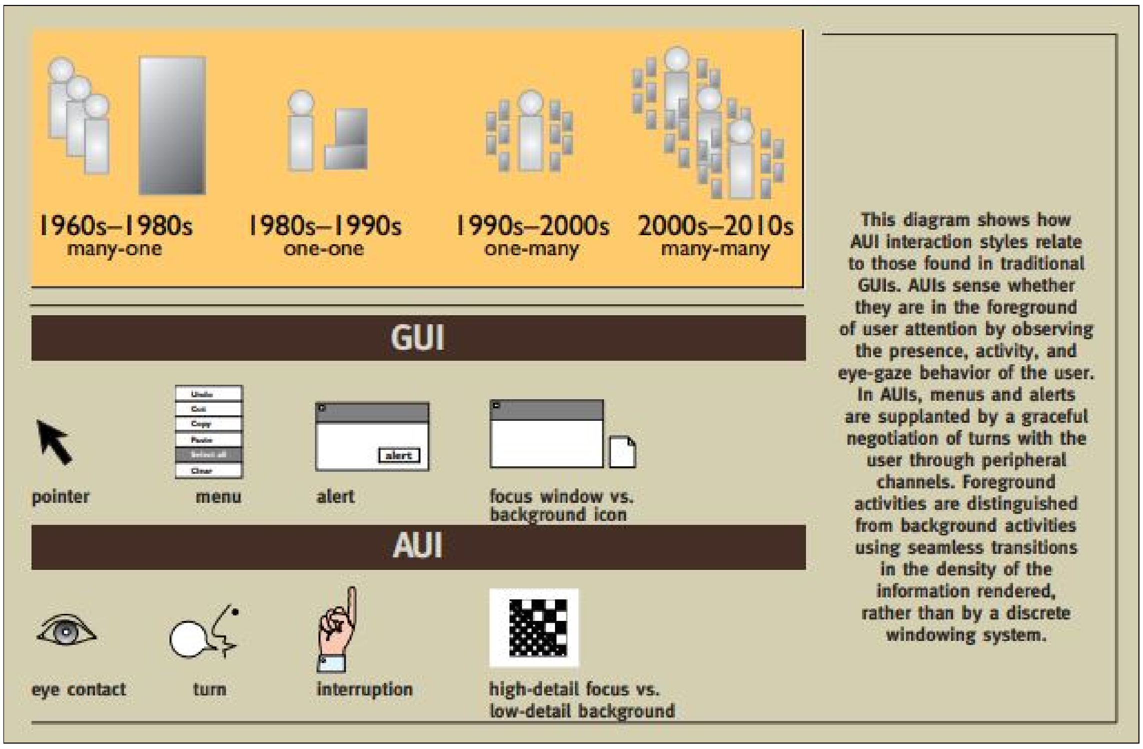
The "ground" of the interaction is defined as whether the interaction is dine on the "foreground" of the human consciousness or the "background". Foreground tasks are defined as those where the user's attention is focused on the target and makes direct effort to communicate with them. Background tasks on the other hand are those which the user is aware are happening, but the user did not make a direct effort to notice, e.g. being aware someone is talking on the phone on the next room or being aware an automatic light was turned on when entering a room but making no direct interaction to light it using a switch (which would be a foreground act).

The target of the communication is either another human or an automated computer system. Human-Human interactions are tasks where the user communicates with another human through a computer system such as a video call or an email while Human-Computer interactions are those where the user communicates with a computer system such as running a program through a Graphical User Interface (GUI).

Buxton states that the importance of the proposed model lies not in simply classifying interactions into one of the four quadrants, but enabling seamless transitions between them based on the context of the user's actions. For example, if the user wants to start a telecommunication with another human but they are not directly available, they should be able to "setup a call when available" which would run on the background and transition to the foreground when the call is accepted by the other side. Similarly, a computer application may monitor the environment and update it's own state while running on the background without directly catching the attention of the user if it is on the background of their awareness.

3.9 Context-aware and adaptive AR

Nowadays, the increase in virtual interfaces on multiple devices has become the norm and has allowed us to monitor multiple systems through virtual User Interfaces (UIs) and increase productivity. But, this increase has also led to the realisation that different UIs compete for the user's attention, ultimately leading to distracting them constantly [64]. Vertegall proposed the implementation of Attentive User Interfaces (AUIs) which aim to focus the user on their task instead of distracting them (see Fig. 52).



This diagram shows how AUI interaction styles relate to those found in traditional GUIs. AUIs sense whether they are in the foreground of user attention by observing the presence, activity, and eye-gaze behavior of the user. In AUIs, menus and alerts are supplanted by a graceful negotiation of turns with the user through peripheral channels. Foreground activities are distinguished from background activities using seamless transitions in the density of the information rendered, rather than by a discrete windowing system.

Figure 52: Attentive UI interface design [64].

AUI gathers inputs passively to determine the user's attention and then ranks virtual communications based on importance according to the active task and then presents the most appropriate information using a turn-taking system by showing information progressively when they capture the user's attention. AUI does not need to fully predict the user's intentions to achieve this, instead the user can be prompted by a simple signal on their periphery and only when they respond to the given signal the virtual information is presented. Such interfaces are also known as "Attention-based" or "Awareness-based" systems.

In addition, AUI can also use the user's capacity for attention to further optimise the interfaces. Analogous to the cocktail party effect, where the user can focus on one active speaker amongst a noisy environment, AUI can enhance the information presented based on the user's attention. Perceptual UIs are one form of AUIs that aim to optimise human-computer interactions through verbal and nonverbal communications. Similarly, Context-aware UIs optimise interactions through providing the most relevant information to the users. The problem with these interfaces lies mostly in trying to model and understand the user's intentions based on their actions and their context as well as finding an optimal way to present such information.

One solution to retrieve context is the use of eye tracking. McNamara et al. [65] utilise eye tracking to introduce spatial and temporal context to augmented information, allowing the system to show only the most relevant augmented interfaces (Fig. 53).

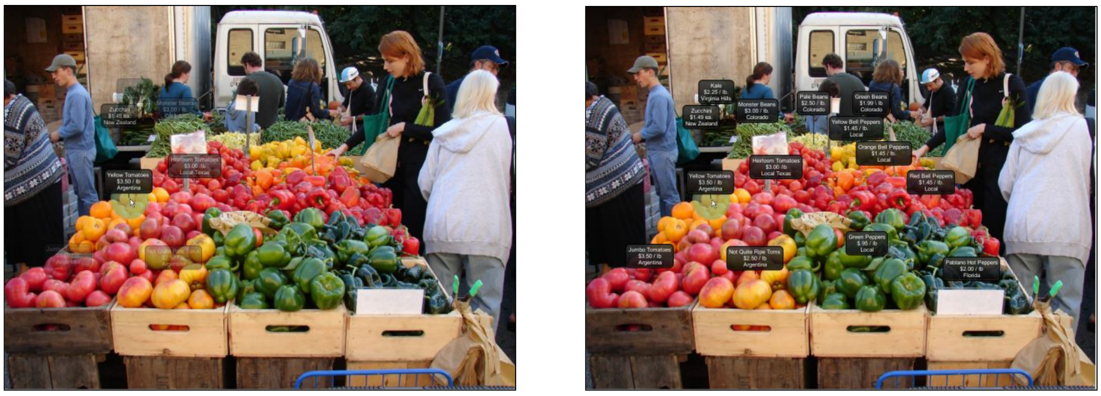

(a) (b)

Figure 53: Information placement with (a) and without (b) spatial context [65]

The results when comparing context-based and context-less showed users made fewer errors in information retrieval when using context-based visualisation compared to context-less. Further studies [66] showed that in more noisy environments (e.g. a supermarket aisles) context-less presentation proved to be more accurate in information retrieval with faster response times and higher accuracy rates, but the experiment was performed in a VR simulation where the limited FOV and performing a real world task are omitted.

Davari et al. [67] adapted virtual interfaces by repositioning or adjusting the transparency of virtual interfaces to allow the user to focus on two independent tasks simultaneously, one on the real world and one on the virtual objects (Fig. 54). During the user study, participants intuitively tended to focus primarily on the real world task and leaving viewing virtual information as secondary, even when asked to do the opposite. Out of the two interface adjustment techniques proposed, participants preferred adjusting the transparency of virtual interfaces rather than moving them, likely due to increased task load to relocate interfaces when they were moved and increase in head movements to view virtual interfaces.

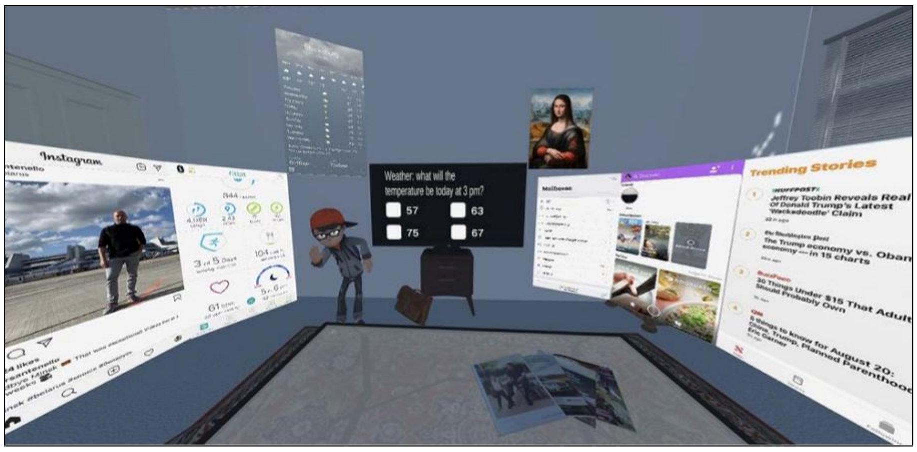
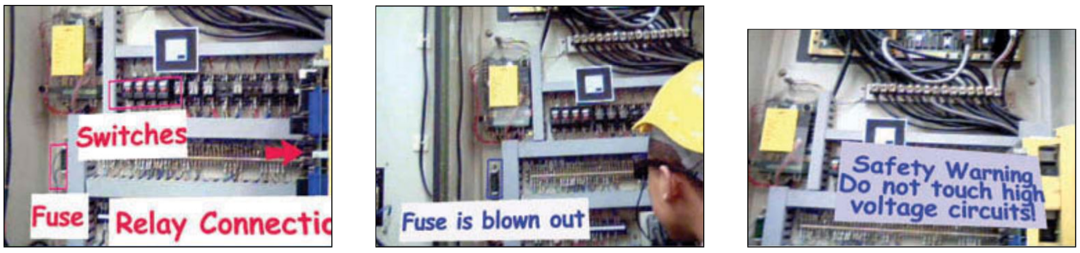


Figure 54: Adaptive virtual interfaces for use while performing real world tasks [67].

Context-aware interfaces have also shown promise in manufacturing guidance when compared to training in VR or traditional paper-based guidance [68]. Zhu et al. designed a system that could adapt the presentation, including color and transparency, as well as content depending on the active task during a maintenance task.



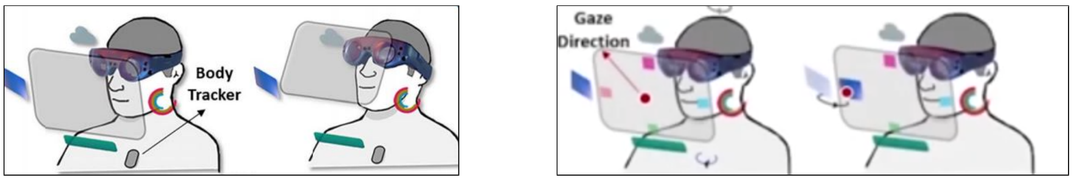
(a) (b) (c)

Figure 55: Context-aware AR maintenance system [68].

Although the system was proven efficient when it was published, compared to newer systems it has some major drawbacks. The context of the application is based on pre-scanned geometry around a black and white marker and each new context requires rebuilding the whole application from scratch including scanning the 3D environment and redesigning the virtual interfaces every time. In addition, the color and transparency were hand picked by the user instead of dynamically adapting based on context. Above all, the interfaces were designed around performing a single task at a time and as such they were placed in a way that occludes multiple real world objects, making it a potential safety hazard.

3.10 Glanceable AR and eye tracking

Glanceable AR is defined as "an interactable paradigm for accessing information in AR, where information resides at the periphery to remain unobtrusive, and it can be accessed at a glance when needed." [14]. Lu et al. proposed two types of Glanceable interfaces, one where virtual information is always visible and can be accessed with a simple eye or head movement and another where virtual objects were "summoned" through user interaction using eye tracking. In both cases, users were able to view the virtual interfaces while following a virtual avatar around the room, proving Glanceable interfaces are able to provide information without distracting the user.



(a) (b)

Figure 56: Glanceable AR. (a) Peripheral interfaces. (b) Gaze-summoned interfaces

Further studies proved the effectiveness of Glanceable interfaces which can be anchored on the real world for stationary tasks and also follow the user when needed for mobile tasks [69]. The combination of Glanceable interfaces with context-awareness showed faster information retrieval in social contexts, such as when conversing with another person that requests some information that can be obtained from the virtual interfaces e.g. the weather [70]. Glanceable AR also tends to be more socially acceptable than other approaches such as using a smartphone during a conversation, due to the interactions being done either indirectly or through eye motions.

Lindbauer et al. went one step further by combining the concepts of Glanceable and context-aware interfaces. The combination of Glanceable and context-aware interfaces can lead to adaptive interfaces that adjust both their visuals and their content based on the user's cognitive load [17]. By detecting the activity the user is performing, the adaptable system adjust the quality of virtual interfaces present, showing more virtual interfaces as the cognitive load is lower (Fig. 57). When the cognitive load increases, the system selects the most relevant interfaces and hides irrelevant information to reduce potential distractions.

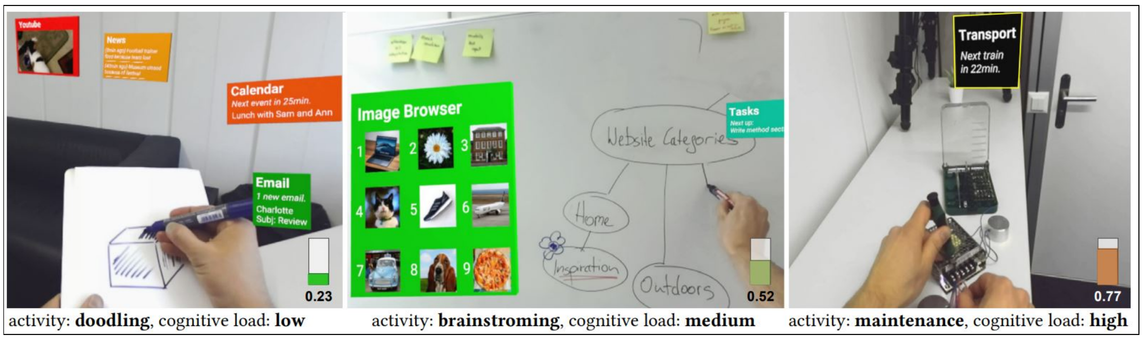


Figure 57: Adaptive interfaces based on cognitive load [17].

By utilising spatial data, the system can reposition interfaces in a way that minimises real world occlusion and places relevant information closer to the user's center of attention. Furthermore, by designing interfaces in multiple Levels of Detail (LODs) (Fig. 58), the system can expand relevant interfaces, showing additional information when necessary.

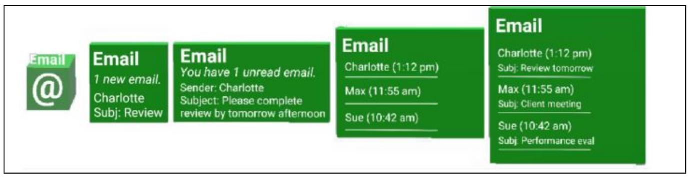
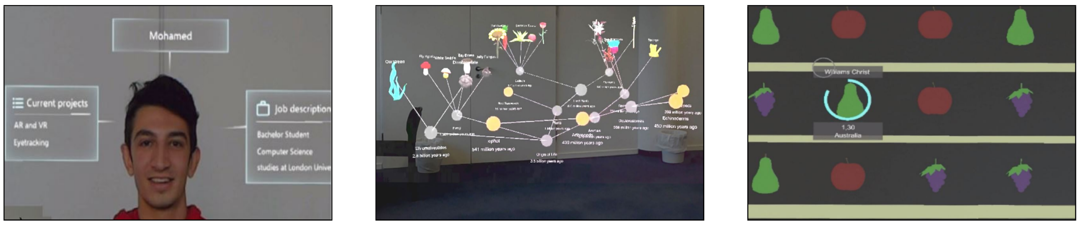


Figure 58: An Email interface designed in 5 Levels of Detail (LODs) [17].

Through their experiments, Lindbauer et al. noticed that users often memorised the positions of the interfaces and instinctively looked towards their last known position, thus repositioning the interfaces actually increased cognitive load. In addition, while using multiple LODs was favorable, users preferred when the LODs were selected based on their input, either directly by interacting with them or indirectly when their gaze was directed towards the interface, rather than when the LOD was automatically selected based on context.

Pfeufer et al. validated the effectiveness of Glanceable, gaze-adaptive interfaces in three distinct scenarios [71]. First, augmented User Interfaces (UIs) were overlayed during a conversation around the speaker's head providing additional context to the conversation (Fig. 59a). The virtual UIs expanded when the user looked towards them and shrank when the user's gaze returned to the speaker. Second, a virtual tree of life was presented dynamically and the user could expand its branches as they pleased using eye tracking interactions (Fig. 59b). Third, a virtual shopping shelf was shown in front of the users who had to identify and select the correct products according to their shopping list using eye tracking interactions (Fig. 59c).



(a) Conversational UI (b) Tree of Life (c) Shopping Shelf

Figure 59: ARTention, Gaze-adaptive Glanceable interfaces in three scenarios [71].

These three scenarios were specifically chosen as the use cases of each differed significantly. On the conversational UI the goal is to keep the focus of the user on the real world, on the tree of life on the virtual content, while the shopping shelf was a mixture of both. In addition, the conversational UI and the tree of life focused more on viewing virtual information while the shopping shelf required both viewing information and interacting correctly with virtual objects.

Results showed positive feedback when viewing information from the virtual objects, but participants faced difficulties when interacting with virtual objects. All interactions followed the gaze-dwell principle, where the user looks at a specific object for a few seconds in order to interact with it. On the tree of life, which consisted of multiple small interactable objects multiple nodes were often in close proximity with each other and the system could not correctly determine the correct focal point causing misinteractions. On the shopping shelf, participants were required to look at each product's labels to identify the correct ones, but due to the nature of the gaze-dwell interactions they often interacted with the same objects after a small delay. The latter issue is commonly known as the "Midas Touch" problem, where an eye tracking system cannot differentiate between "viewing" and "interacting" with virtual objects. Furthermore, when tested with different gaze-dwell delay times to find the optimal time for interactions, the results showed that there is no globally acceptable value and it is purely personal preference.

Based on the aforementioned results, we can surmise that the effectiveness of a Glanceable AR system significantly depends on the accuracy of the underlying eye tracking system. While one solution is to simply wait until eye trackers improve over time on a hardware level, it is equally important to optimise virtual content in order to minimise the necessary accuracy required for a system to be usable without significant issues.

With the combination of marker-based detection with the Magic Leap's integrated eye tracking Jing et al. [72] showed that in a collaborative scenario even when the users had a negative view of the eye tracker's accuracy when viewing their eye tracking cursors (with an offset around 3-5 cm, which was similar to the distance between the markers of the experiment), they were still able to collaborate more efficiently when they both saw each others' cursors (Bi-directional collaboration) when they used eye tracking than when they used hand tracking to move their cursors.

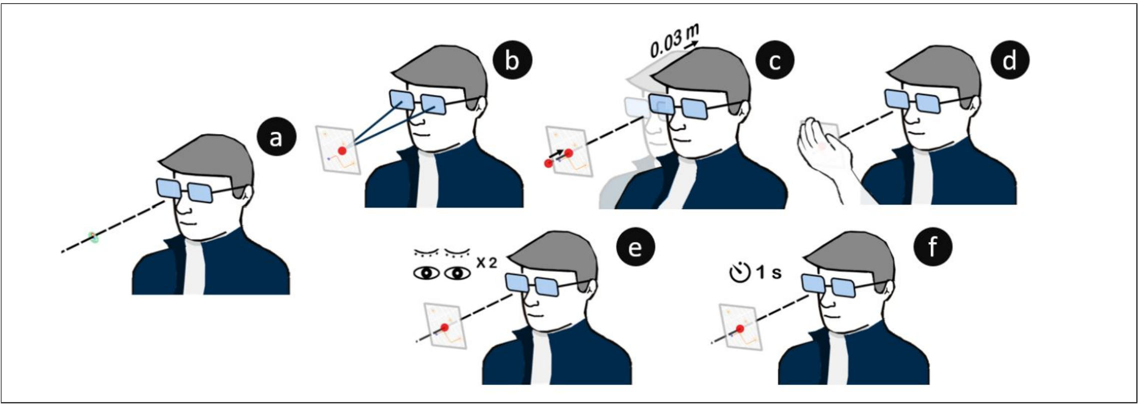


Figure 60: Different eye tracking interactions for AR [73]

Lu et al. [73] on the other hand explored the effectiveness of different interaction types in Glanceable AR that primarily rely on eye tracking (Fig. 60). In their work, they evaluated five different interaction techniques on two tasks: one walking task and one sitting. The five interaction techniques include:

- **Fixation glance** (FG) (Fig. 60 b) - Converge binocular gaze at the depth of the interface.
- **Head-depth** (HD) (Fig. 60 c) - Focus at an interface and lean backwards 3mm to interact.
- **Hand-Overlay** (HO) (Fig. 60 d) - Place hand slightly behind the virtual object.
- **Blink** (BL) (Fig. 60 e) - Blink twice to interact.
- **Gaze-Dwell** (DW) (Fig. 60 f) - Position gaze on a virtual object for 1 second to interact.

Results showed no clear superior choice among the five with different interactions being viable based on the scenario. When ranking each interaction based on preference, HD was chosen as the most preferred interaction followed by DW and BL on the walking task, while FG was the most preferred interaction followed by BL and DW on the sitting task with HD being the most disliked. In terms of false activation rates, DW has by far the highest false activations with rates of over 70% in both scenarios, followed by FG. HO and HG had the lowest false activation rates at approximately 10% while BL was slightly higher in both scenarios. Finally, in terms of social acceptance BL was the preferred interaction type in all social scenarios followed by DW with the other three having significantly lower social acceptance.

In summary, Gaze-Dwell and Blink-based interactions seem to be the most widely accepted interactions across different scenarios and social acceptance. While other interactions still have merits, such as a lower false activation rate or are preferred in different scenarios, Gaze-Dwell and Blink-based interactions are widely accepted in both walking and sitting tasks in addition to being more socially acceptable.

4 Machine shop training

In this chapter we will analyze our first work in designing a machine shop education environment using a serious game [2]. The serious game was developed with guidance from the Micromachining and Manufacturing Modeling Lab (M3 lab) of the School of Production Engineering an Management of the Technical University of Crete, supervised by Prof. Aristomenis Antoniadis. Development of the Serious game was accomplished by Grigorios Daskalogrigorakis, Salva Kirakosian and Prof. Katerina Mania of Team Surreal of the school of Electrical and Computer Engineering of the Technical University of Crete.

The M3 lab provided the 3D object models, 2D documentation in addition to theoretical guidance regarding manufacturing processes and manufacturing education. Salva kirakosian was responsible for 2D and 3D design for the 3D virtual machine shop, while Grigorios Daskalogrigorakis was the gameplay designer and developer regarding the Gamification of the manufacturing process.

The serious game was used as an educational tool for teaching G-code programming remotely on a 5th year course of the school of Production engineering and Management for over 150 students during the pandemic in the year 2021.The serious game served as a first step in streamlining manufacturing tasks in a streamlined format as gamified missions. These gamified missions were later optimised for use in AR guidance using Holo box.

4.1 Motivation

Training new personnel for Computer Numerical Control (CNC) manufacturing is often a tedious operation. Practice under real-world machinery is expensive, consuming vital system's resources and, in some cases, dangerous for the trainee. G-code programs guide the machining process, written in a computer or machine controller and loaded to a machine. A safer alternative for training uses simulated factories or training environments.

The concept of a digital factory can be defined as an interactive 3D environment supporting modeling, communications and operation of manufacturing. Previous studies indicate that virtual training is effective as the trainee is guided through complex product manufacturing. While there are plenty of gamified virtual simulations that aid in manufacturing training and mainly focus on safety tasks or manual operation, the task of creating a serious game focused on writing G-code is technically challenging. Potential trainees often have no previous programming knowledge. It is significant that they learn how to write efficient G-code programs, not just correct ones.

This works presents an innovative desktop-based CNC machining training procedure, created as a serious game for CNC manufacturing training. The proposed serious game prepares the trainee in writing G-code and setting up virtual machines for completing milling and turning tasks. The serious game environment offers a complete training procedure, based on the flow of a modern game. The trainee is offered tutorials and step by step guidance and is then asked to complete further manufacturing tasks using the knowledge acquired. Based on the target audience, the trainee can learn either milling or turning procedures independently. Our training procedure works as a standalone training system without the need for a supervisor or trainer to be present.

The trainee is assigned to write the correct G-code while undertaking turning or milling missions known as tasks. The trainee reads the provided documentation and the mission specifications and then sets up the milling or turning machinery with the appropriate cutting tools and manufacturing materials for each mission. The game presented is an introductory step towards innovative CNC manufacturing training for trainees with minimal or no prior knowledge, such as bachelor engineering students without an engineering degree. They are going to evolve into a new generation of technically proficient manufacturing workers.

4.2 The manufacturing process

Controlling a real-world CNC machine is possible through three methods. The simplest one is the use of a Numerical Control (NC) unit, which allows for manual control through the use of an interface of buttons and controls. The second method is the use of commands written in a predefined format named "G-code". G-code programs are written on a computer or machine controller and loaded to a machine. The final method is the use of Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) software that convert a virtual 3D design into G-code. Among these three methods, using an NC unit is optimal for simple tasks, while CAM programs are used by 3D designers and not average users. G-code is a simple and usable low-level command set for the creation of both simple and complex objects.

By using G-code, an operator instructs a machine how to manufacture a desired object. This involves instructing the machine in relation to which motors to move, where to move and what path to follow. The most common use cases include cutting jobs, such as milling or cutting where a tool or cutting material is moved through a tool path cutting away unwanted material while leaving the desired work piece. Similarly, instead of cutting tools the G-code can also be used in additive manufacturing machines, such as in 3D printers that photo-plot objects instead of cutting.

Milling and turning are two of the most widely used machinery in manufacturing. Turning machines use a non-rotary cutting tool that moves more or less linearly along 1 to 3 axes of motion while cutting a rotating stock material. Cutting can be performed on the outside or the inside of the material to produce tubular components of various geometries. Milling on the other hand advances a rotary cutting tool into a stock material in varying directions on several axes and with varying feed rate.

4.3 Traditional manufacturing education

In order to design an innovative game for efficient machining training, an extensive interview obtaining feedback concerning machining educational processes was conducted involving university faculty in the M3 lab, where undergraduate and postgraduate students are educated in the field of Mechanical Engineering. During this interview, information regarding standard teaching methods was obtained involving theory and documentation and the limitations of education through software-based CAD-CAM simulation were discussed.

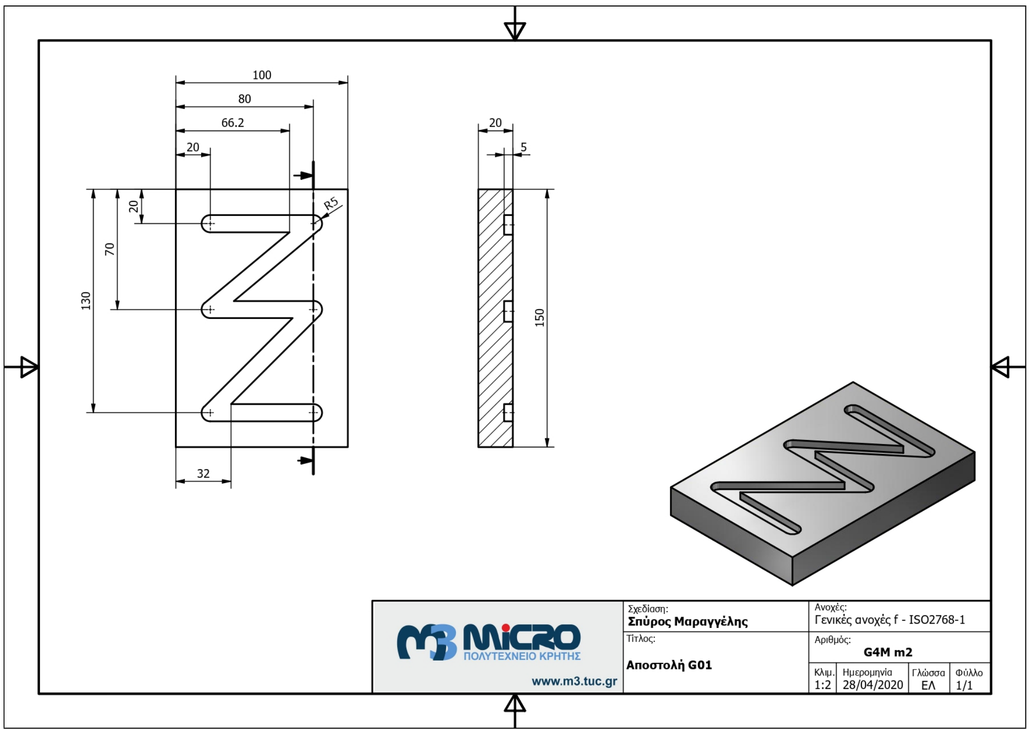


Figure 61: Milling mechanical blueprint

Researchers in the lab described the traditional hands-on procedure of training undergraduates using G-code and CNC machines, where each student performs a hands-on operation to manufacture a requested piece on a milling or turning machine. The manufacturing task is performed by following a mechanical blueprint (see Fig. 61) to produce a finished product with guidance from a supervisor.

As the number of students entering university education in machining exceeds 150 every year and the number of milling and turning machinery available is limited, real-world training in manufacturing is limited. In addition, the risk of a workplace accident from an inexperienced student is high, thus every student requires constant supervision. In most cases, during the course of a machining lesson, the students use simulators to write G-code. Simulators, though, lack the sense of an actual machine shop and students are often overwhelmed when performing a task on actual machinery. In addition, we obtained hands-on experience inside an actual machine shop in the manufacturing floor of the M3 lab.

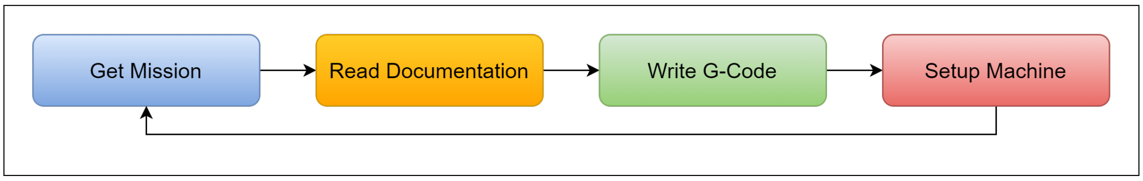


Figure 62: The training procedure

After the interview, a brainstorming session was held where we streamlined the steps of the manufacturing process in a gamified scenario (see Fig. 62). Overall, we can simplify the process of manufacturing in three distinct steps: Receiving a manufacturing request from a client or professor with the appropriate data, writing the G-code that produces the requested object and finally adding the appropriate cutting tool and cutting material to the machine. While there are other steps such as performing simple maintenance to a malfunctioning machine or cleaning the machine after each use, we decided to omit such side steps for the sake of simplicity and focus on the educational processes involving these three distinct steps.

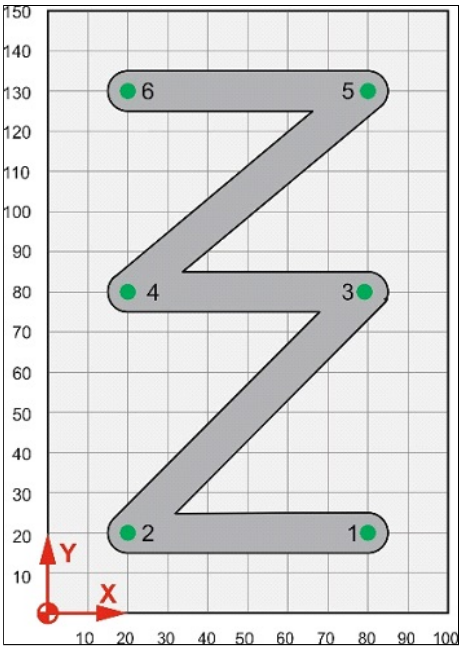
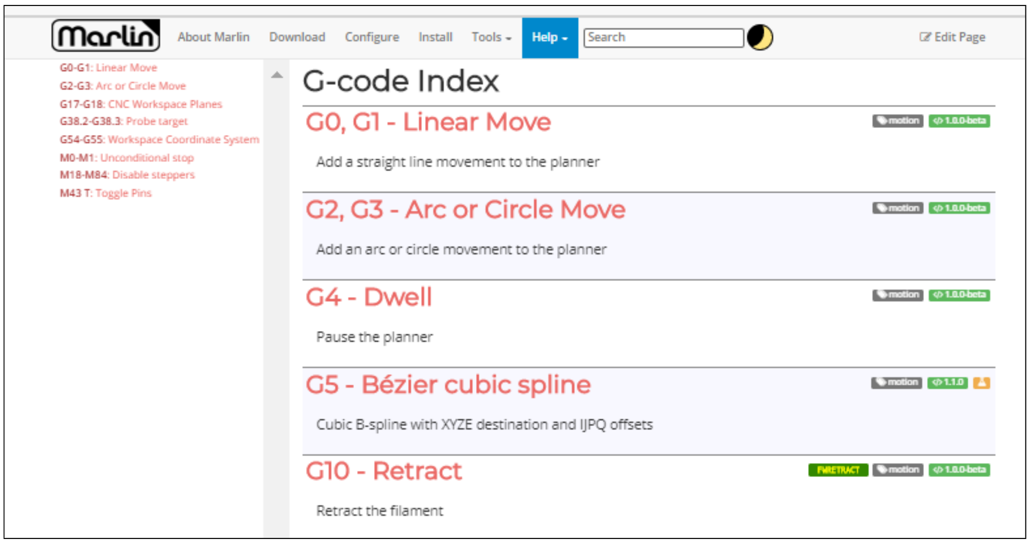


Figure 63: Cutting path diagram

When a client or professor in a machining laboratory requests an object, they provide the employee or trainee with the shape of the requested object, the material and the relevant quantity. The shape of the object is described in a 2D mechanical drawing (see Fig. 61) depicting a representation of the object through views and providing the required dimensions. The material of the object is extracted from the constraints imposed by the object's usage, e.g. tension, elasticity, heat resistance etc, but the appropriate material and quantity is often chosen by the client. Based on the requested material and blueprint details, the employee or trainee must determine the initial parameters of the cutting procedure such as the maximum allowed spindle speed or maximum feed rate, which are provided by the tool manufacturers for all given tools and materials. Next, the mechanical drawing is translated by the employee into G-code, which consists of simple motions, such as a sequence of straight line commands or circular commands (see Fig. 63). For more complex objects a CAM software is required.



G0-G1: Linear Move
G2-G3: Arc or Circle Move
G17-G18: CNC Workspace Planes
G38.2-G38.3: Probe target
G54-G55: Workspace Coordinate System
M0-M1: Unconditional stop
M18-M84: Disable steppers
M43 T: Toggle Pins

G-code Index

G0, G1 - Linear Move — motion, 1.0.0-beta
Add a straight line movement to the planner

G2, G3 - Arc or Circle Move — motion, 1.0.0-beta
Add an arc or circle movement to the planner

G4 - Dwell — motion, 1.0.0-beta
Pause the planner

G5 - Bézier cubic spline — motion, 1.1.0
Cubic B-spline with XYZE destination and IJPQ offsets

G10 - Retract — FWRETRACT, motion, 1.0.0-beta
Retract the filament

Figure 64: G-code documentation

After the cutting path is determined, the G-code programming step is the translation of each step of the execution process to the appropriate G-code commands. Since each G-code command requires different parameters and is a low level programming language, the process is a tedious step by step lookup through a manual (see Fig. 69) so that the correct syntax for each command is determined, becoming easier through practice and memorisation. The evaluation of the G-code is accomplished through CAM software such as Siemens NX, which simulates the cutting procedure to check its correctness and then translates the code into native cutting motions based on the manufacturing machine's specific model.

Finally, the process is completed by physically adding the appropriate tool and material to the machine and executing the program. The employee then monitors the cutting process making necessary adjustments, such as adding new cutting materials after each cut or adding new cutting tools after they are worn out.

4.4 3D CNC Machine shop

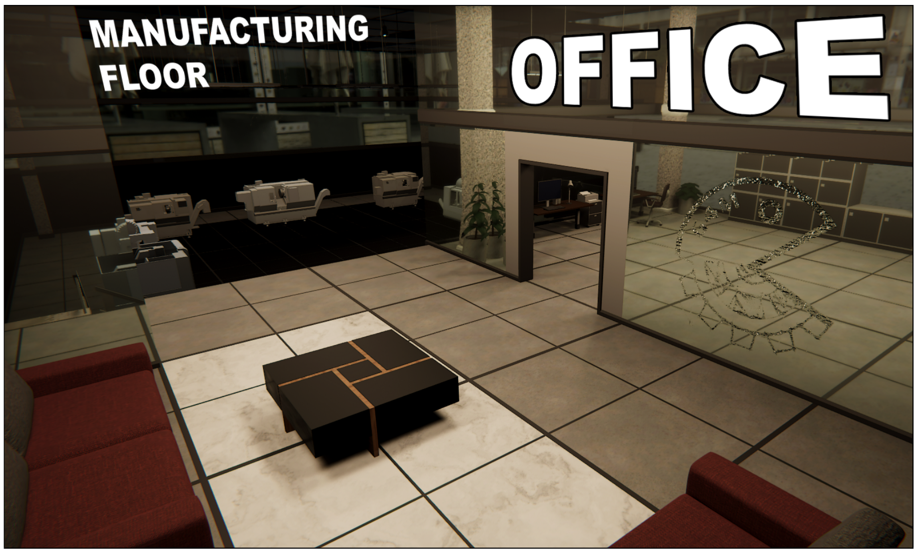


Figure 65: The virtual machine shop

The 3D CNC Machine Shop is the core environment (see Fig. 65). The design of the space was inspired by a real-world machine shop layout and adjusted in scale and look to be more visually appealing to the users. An alternative 3D machine shops was also presented using realistic scale and visuals, but the assistants from the M3 lab preferred the stylised environment that offered stronger visual satisfaction. They expected that visual satisfaction would offer incentive for more 'play time' and, thus, training time without exhausting the trainee.

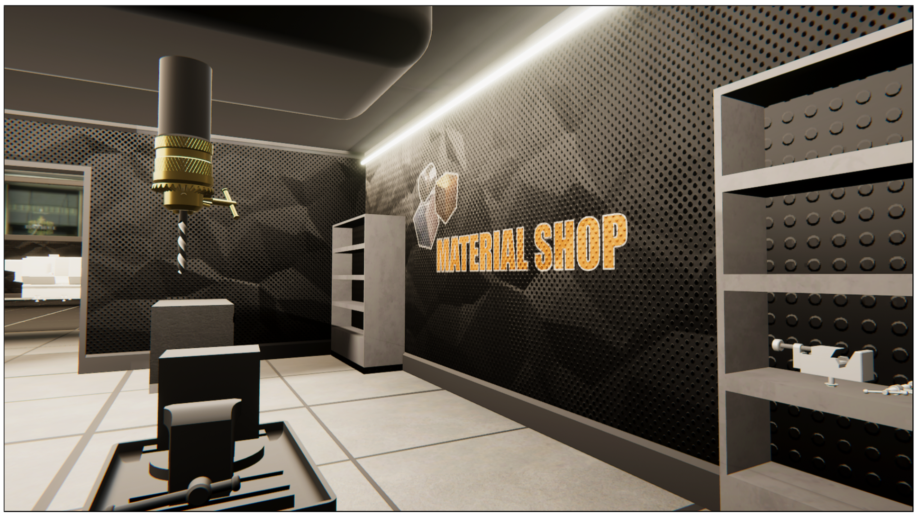
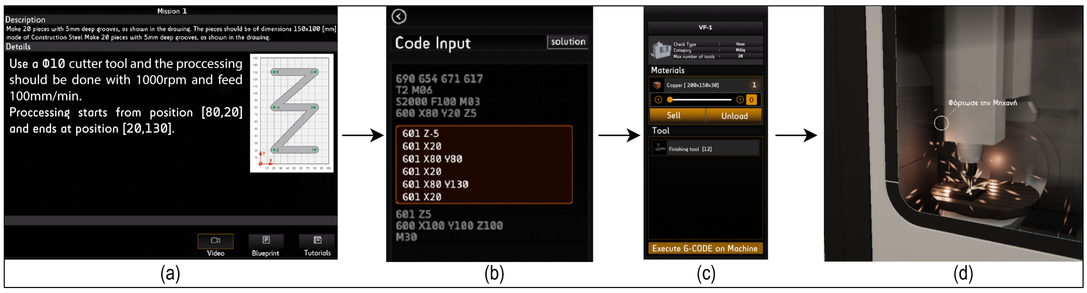


Figure 66: The virtual storage and shop

The virtual machine shop comprises of three key areas required for gameplay. The first is an office inside which the trainee can get machining training missions and write G-code on a virtual computer. The second area is the main manufacturing floor where the milling and turning machines reside. The third area is the storage, inside which the trainee can go and retrieve cutting tools and materials (see Fig. 66). Additional areas were filled with secondary objects such as a lounging area or storage cabinets that are not essential to the manufacturing process but result in a highly compelling and realistic gaming experience.

During the course of the training process, the trainee roams around the machine shop and interacts with specific User Interfaces (UIs) allowing reading documentation, obtaining new missions, writing the G-code, buying new tools and materials as well as slotting tools and materials inside machines.

4.5 The training procedure


Mission 1

Description

Make 20 pieces with 5mm deep grooves, as shown in the drawing. The pieces should be of dimensions 150x100 [mm] made of Construction Steel Make 20 pieces with 5mm deep grooves, as shown in the drawing.

Details

Use a Φ10 cutter tool and the proccessing should be done with 1000rpm and feed 100mm/min.
Proccessing starts from position [80,20] and ends at position [20,130].

Code Input

```
G90 G54 G71 G17
T2 M06
S2000 F100 M03
G00 X80 Y20 Z5
 G01 Z-5
 G01 X20
 G01 X80 Y80
 G01 X20
 G01 X80 Y130
 G01 X20
G01 Z5
G00 X100 Y100 Z100
M30
```

VF-1

Check Type : Haas
Category : Millig
Max number of tools : 20

Materials

Copper [200x150x30] 1

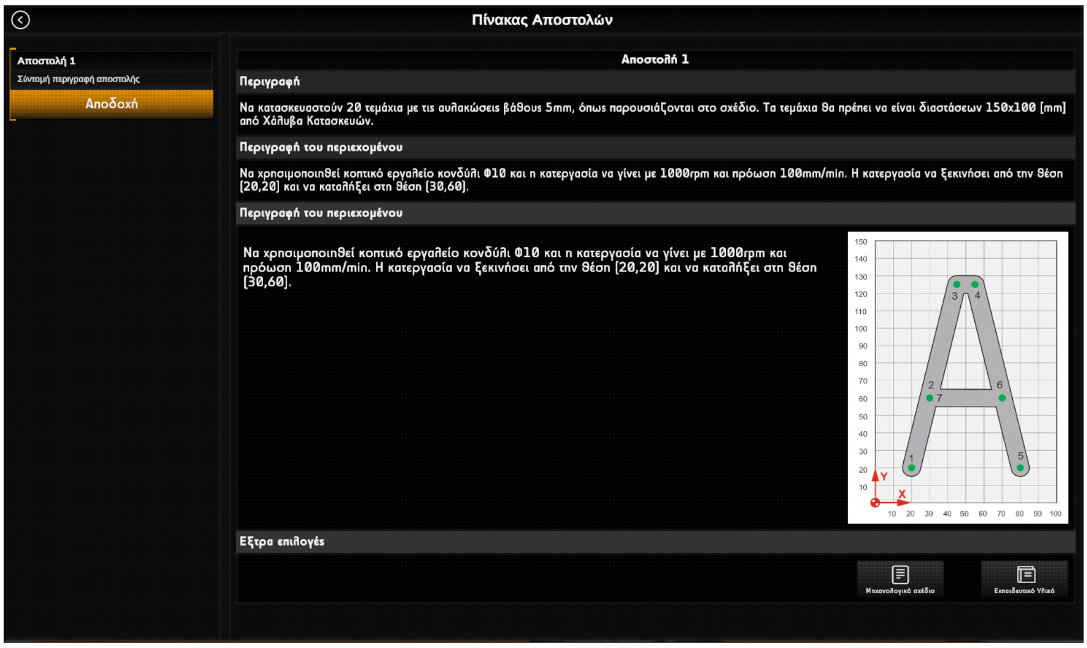
Tool

Finishing tool [12]

Figure 67: An overview of the training procedure proposed. The player accepts a mission (a), writes the correct G-code (b), loads cutting tools and materials on the machine (c) and finally cuts the requested object. The result of the final object is then shown inside the milling or turning machine (d).

The training starts from either turning or milling processes. The trainee selects to start with either a turning machine or a milling machine at the start of the game. They can unlock more machines of the other type during the course of the game if they desire. Initially, the trainees are given instructions on how to play the game and interact with virtual objects. The first time the trainees enter the training environment, they are greeted by a tutorial that introduces the basic mechanics of the game. An animated tour of the virtual machine shop is presented including a short explanation of each key component of the factory as an engaging virtual tour of the factory. The main machining tutorials follow including the first two G-code commands required to complete the first missions. After completing the tutorial, the trainee accepts new training missions instructing to manufacture various objects utilizing the G-code commands learnt.

After each mission is completed, the trainee is awarded with virtual "Experience" and "Gold". The trainee carries on completing missions of increasing difficulty based on their "Experience". They can choose to unlock more machines of a different type, for example a new milling machine or a new turning machine in order to continue their training by using "Gold". Each mission is comprised of four distinct steps (see Fig. 62). The trainee initially accepts a new mission which contains the request to manufacture a specific object. The trainee then proceeds to read any appropriate documentation and write the correct G-code that produces the requested object. When the written code is correct the trainee proceeds to the storage in order to purchase the appropriate cutting tools and materials and then load them to the milling or turning machine on the manufacturing floor.


Πίνακας Αποστολών

Αποστολή 1
Σύντομή περιγραφή αποστολής

Αποστολή 1

Περιγραφή

Να κατασκευαστούν 20 τεμάχια με τις αυλακώσεις βάθους 5mm, όπως παρουσιάζονται στο σχέδιο. Τα τεμάχια θα πρέπει να είναι διαστάσεων 150x100 [mm] από Χάλυβα Κατασκευών.

Περιγραφή του περιεχομένου

Να χρησιμοποιηθεί κοπτικό εργαλείο κονδύλι Φ10 και η κατεργασία να γίνει με 1000rpm και πρόωση 100mm/min. Η κατεργασία να ξεκινήσει από την θέση [20,20] και να καταλήξει στη θέση [30,60].

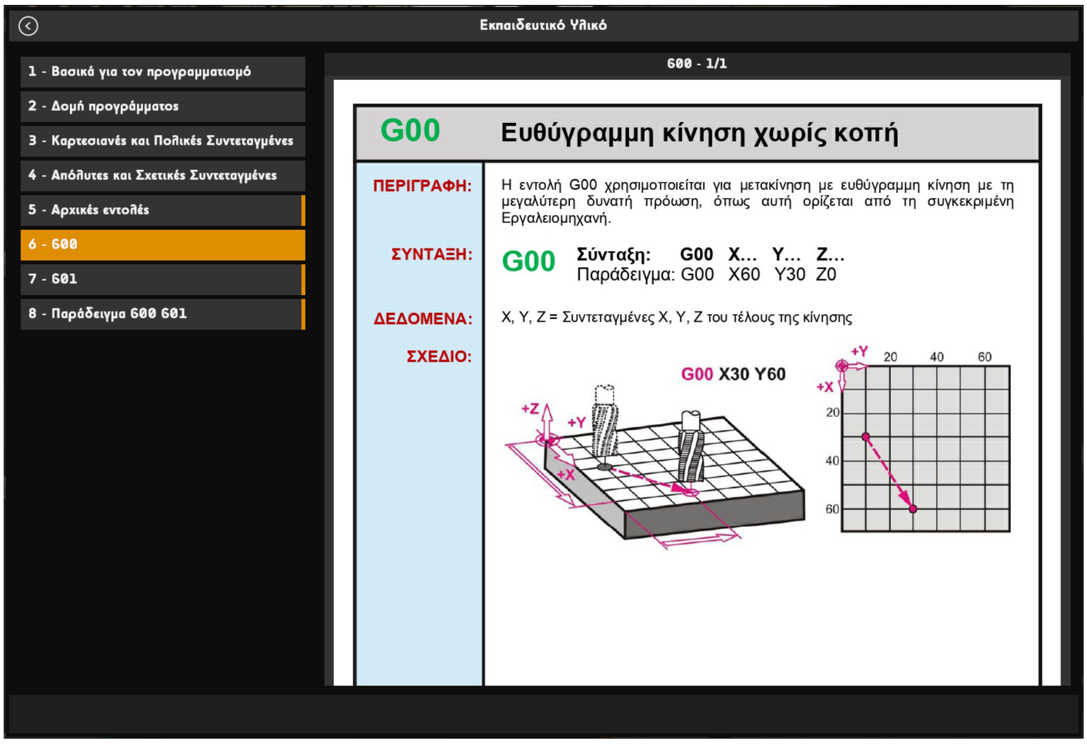
Περιγραφή του περιεχομένου

Να χρησιμοποιηθεί κοπτικό εργαλείο κονδύλι Φ10 και η κατεργασία να γίνει με 1000rpm και πρόωση 100mm/min. Η κατεργασία να ξεκινήσει από την θέση [20,20] και να καταλήξει στη θέση [30,60].

Εξτρα επιλογές

Figure 68: The Mission Board interface.

Get mission. The trainee accepts a new mission through the mission board interface (see Fig. 68) by reading the mission description accompanied by the cutting path blueprint (see Fig. 67 a). The mission description is also accompanied by the mechanical drawing (see Fig. 61) which can be viewed by the respective button on the same interface. Through the same interface, the trainee can also view the documentation interface through a second button.


Εκπαιδευτικό Υλικό

1 - Βασικά για τον προγραμματισμό
2 - Δομή προγράμματος
3 - Καρτεσιανές και Πολικές Συντεταγμένες
4 - Απόλυτες και Σχετικές Συντεταγμένες
5 - Αρχικές εντολές
6 - G00
7 - G01
8 - Παράδειγμα G00 G01

G00 - 1/1

G00 **Ευθύγραμμη κίνηση χωρίς κοπή**

ΠΕΡΙΓΡΑΦΗ: Η εντολή G00 χρησιμοποιείται για μετακίνηση με ευθύγραμμη κίνηση με τη μεγαλύτερη δυνατή πρόωση, όπως αυτή ορίζεται από τη συγκεκριμένη Εργαλειομηχανή.

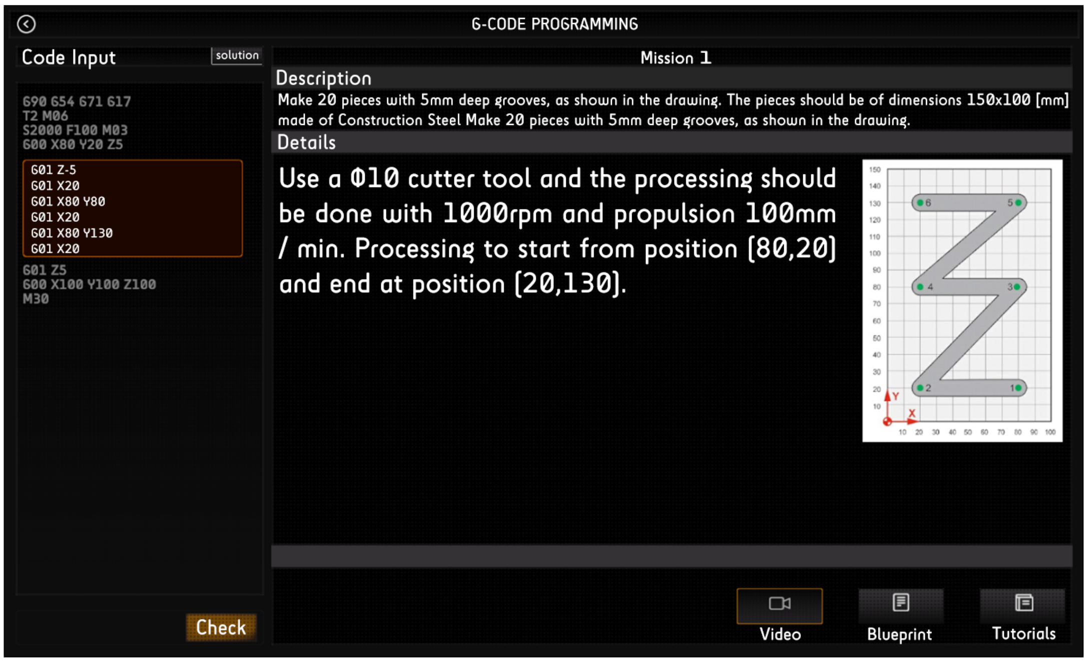
ΣΥΝΤΑΞΗ: **G00** **Σύνταξη:** **G00 X... Y... Z...**
Παράδειγμα: G00 X60 Y30 Z0

ΔΕΔΟΜΕΝΑ: X, Y, Z = Συντεταγμένες X, Y, Z του τέλους της κίνησης

ΣΧΕΔΙΟ:

Figure 69: The documentation interface.

Read documentation The documentation interface contains pages from a G-code textbook as authored by faculty of the university machining laboratory, used as a reference material for undergraduate students (fig. 69). The documentation contains a selection of tabs, each providing documentation regarding the general operation of the machines, a specific G-code command syntax or a full example of a specific G-code command. The pages shown in the documentation tab are filtered, showing only the pages including related information for a requested mission while also highlighting any unread material, therefore, helping the students focus and understand the necessary documentation without having to search for it in an effort to reduce frustration.


G-CODE PROGRAMMING

Code Input | solution

```
G90 G54 G71 G17
T2 M06
S2000 F100 M03
G00 X80 Y20 Z5
G01 Z-5
G01 X20
G01 X80 Y80
G01 X20
G01 X80 Y130
G01 X20
G01 Z5
G00 X100 Y100 Z100
M30
```

Mission 1

Description

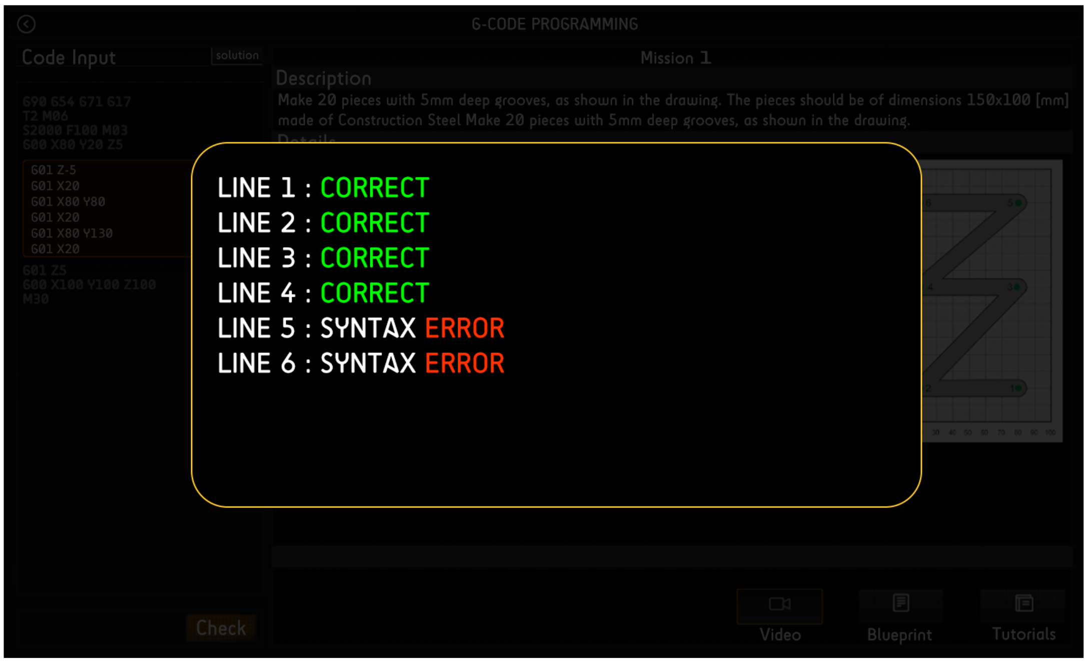
Make 20 pieces with 5mm deep grooves, as shown in the drawing. The pieces should be of dimensions 150x100 [mm] made of Construction Steel Make 20 pieces with 5mm deep grooves, as shown in the drawing.

Details

Use a Ø10 cutter tool and the processing should be done with 1000rpm and propulsion 100mm / min. Processing to start from position (80,20) and end at position (20,130).

Figure 70: The G-code interface. Programming interface including the G-code input field on the left and mission info on the right.

Write G-code. The G-code programming interface includes a code input interface along the mission description interface (see Fig. 70). During prototyping, user feedback communicated that following an optimal tool path shouldn't be optional. It was significant that the trainees learn how to efficiently write G-code, not simply correctly. As is, the initialization and ending of the G-code is pre-written and the trainees are asked to write the central portion of the main cutting process by following a given path blueprint. The only code that is, therefore, checked for correctness is the core code of the main cutting process.



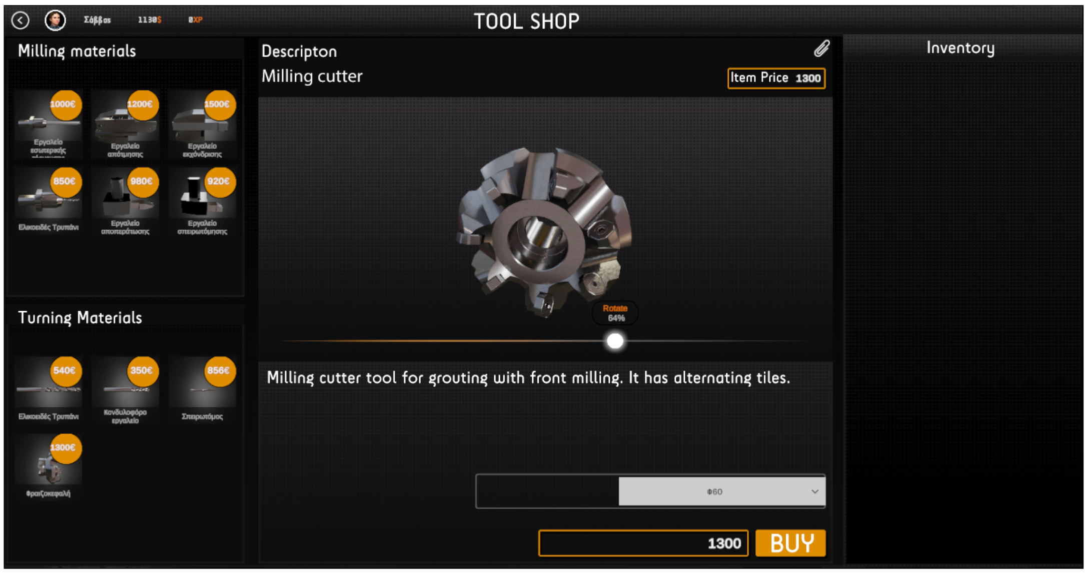
Make 20 pieces with 5mm deep grooves, as shown in the drawing. The pieces should be of dimensions 150x100 [mm] made of Construction Steel Make 20 pieces with 5mm deep grooves, as shown in the drawing.

```
G90 G54 G71 G17
T2 M06
S2000 F100 M03
G00 X80 Y20 Z5
 G01 Z-5
 G01 X20
 G01 X80 Y80
 G01 X20
 G01 X80 Y130
 G01 X20
G01 Z5
G00 X100 Y100 Z100
M30
```

```
LINE 1 : CORRECT
LINE 2 : CORRECT
LINE 3 : CORRECT
LINE 4 : CORRECT
LINE 5 : SYNTAX ERROR
LINE 6 : SYNTAX ERROR
```

Figure 71: The G-code interface feedback after the "Check" button is pressed.

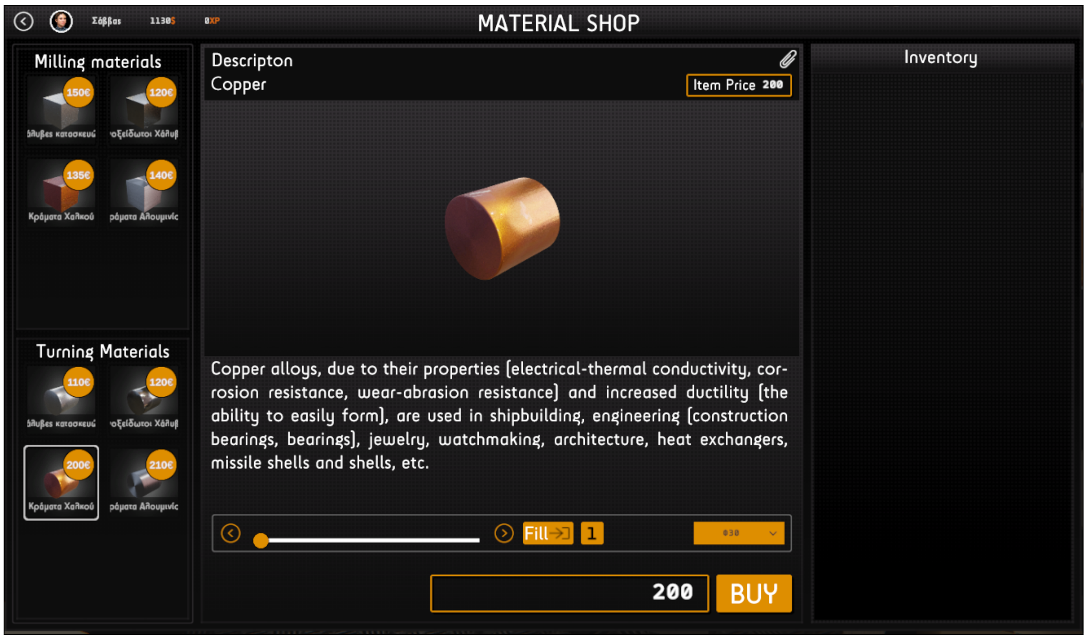
The trainee can check the correctness of the code they have written and receive feedback in relation to potential mistakes (see Fig. 71). The correct cutting path and useful information about each mission are always visible to allow the trainee to view them while writing the G-code. The trainees can try as many times as they need in order to complete the mission without any time limitations or penalty or machine failure as in the real-world.



Milling cutter tool for grouting with front milling. It has alternating tiles.

Figure 72: The tool shop interface. On the left panel various tools are shown, on the central panel there is a visual representation of the selected tool along with a description. On the bottom side there is a buy button with a selection related to the dimensions of the tool.

Machine setup After the trainees have written and corrected their code, the final step is to setup the physical machine for the process. The trainee is asked to add the appropriate cutting tool and stock material on the machine, as stated in the mission description.



Copper alloys, due to their properties (electrical-thermal conductivity, corrosion resistance, wear-abrasion resistance) and increased ductility (the ability to easily form), are used in shipbuilding, engineering (construction bearings, bearings), jewelry, watchmaking, architecture, heat exchangers, missile shells and shells, etc.

Figure 73: The material shop interface. On the left panel various materials are shown, on the central panel there is a visual representation of the selected material along with a description. On the bottom side there is a buy button with a selection related to the dimensions of the material along a quantity slider for buying multiple at once.

The trainee can walk around heading for the storage in order to buy the appropriate cutting tools (see Fig. 72) and materials (see Fig. 74). The trainees, then, interact with the actual CNC machine and "slot" the tools and materials on the machine before executing the code (fig. 67 c). Allowing for the trainee to walk around the machine shop and interact with machinery is not necessary for G-code training but it provides trainees with a higher sense of immersion. Interacting with machinery and receiving minimal feedback while executing a program allows the trainees to feel a higher sense of accomplishment as opposed to displaying a simple congratulatory message.

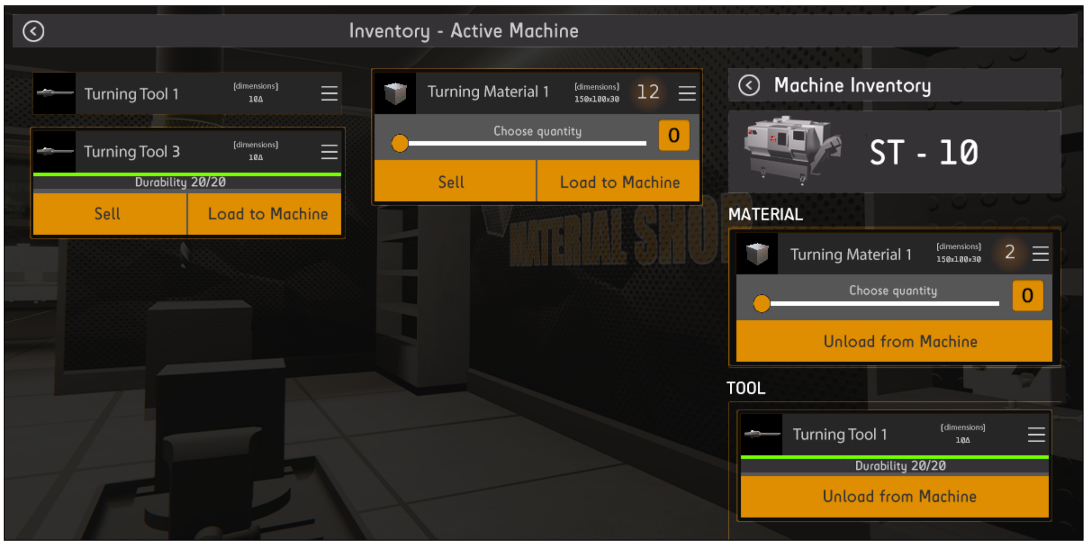


Figure 74: The player and machine inventories. On the left and central column are the player's tool and material inventories. On the right is the machine's inventory. Each tool and material has a button to load it to the machine or unload it from the machine.

4.6 Evaluation and results

The serious game was evaluated on a short-term pilot test. Participants were selected to include 6 experienced users in manufacturing from the M3 lab as well as a class of approximately 150 undergraduate students with minimal to no previous knowledge. The participants were asked to play at their leisure in the span of 2 months and complete at least 13 out of 43 available manufacturing missions. Participants then filled an online questionnaire regarding their experience along with sending their save files for evaluation.

Undergraduate students were also given printed versions of 10 missions and were asked to solve the same missions in AutoCAD and submit their solutions. In addition, they were asked to compare the mental and temporal demand as well as the enjoyment of these two methods.

Participants showed they preferred the virtual environment for multiple reasons. Firstly, participants liked it when they had step-by-step guidance in every step when completing a mission as well as the direct feedback when writing G-code in the serious game compared to running the AutoCAD's simulation and manually comparing the requested blueprint with the extracted 3D model. Secondly, participants favored the streamlined approach given in the serious game's mission compared to the free-form text tasks written by hand. After completing a few missions, the streamlined gamified format of the descriptions made it easier to read and notice subtle differences between missions. Finally, participants were more eager to complete more missions inside the serious game with a vast majority of students competing all 43 available missions. Comparatively, participants mentioned they took 2 to three times as long to solve and check their solutions using an AutoCAD simulation, and even then some students sent incorrect solutions with most students sending fewer than half the solutions.

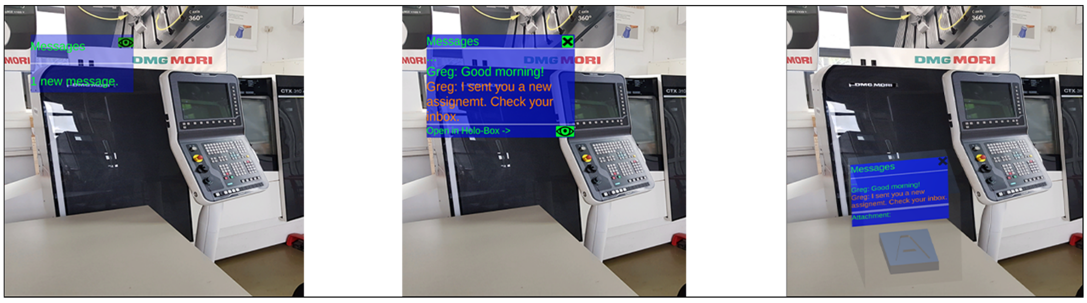
5 Holo-Box: Level-of-Detail Glanceable Interfaces for Augmented Reality

In this chapter we analyze the initial implementation of our work. The main motivation for this work was to combine the gamified mission format presented in G-code Machina [2] with Glanceable AR interfaces [14] in order to provide realtime guidance inside the manufacturing workspace using AR in a streamlined gamified manner.

Although this work had positive feedback when published [3] its usability was limited by issues regarding the Magic Leap's eye tracking accuracy. These issues were identified and corrected during the course of this work for the final implementation.

5.1 3-LOD architecture

Holo-box employs a three Level of Detail (LOD) interface combination, with each LOD revealing varying levels of information with varying levels of real world occlusion. Two of the three LODs are 2D display-fixed interfaces which are always visible at a fixed position of the user's visual field and use eye tracking for interactions with virtual objects. The third interface is a 3D world-based interface which receives hand-based inputs instead.


Messages
1 new message.

Messages
Greg: Good morning!
Greg: I sent you a new assignemt. Check your inbox.
Open in Holo-Box ->

Messages
Greg: Good morning!
Greg: I sent you a new assignemt. Check your inbox.
Attachment:

Figure 75: Holo-Box's 3 Levels of Detail interfaces shown in the context of a machine shop. (left) Simple Glanceable. (middle) Detailed Glanceable. (right) Holo-Box.

Initially, the user is presented with the first LOD interface and the usability of the 3-LOD system is as follows:

Level 1: Simple Glanceable interfaces (SG) (Fig.75 left), are visible by default while the user is focused on the real world, are compact and semi-transparent. Their purpose is to alert users to changes of the virtual content. Interaction is initiated by gaze-activated virtual buttons, using a small gaze-dwell delay. This short gaze-dwell initiates a transition to a Detailed Glanceable (DG) interface.

Level 2: Detailed Glanceable interfaces (DG) (Fig.75 center), provide more complex information compared to SG UIs such as lines of text or images. DG UIs are less transparent than SG. DG elements do not cover the entire FoV allowing for real-world spatial awareness. Using gaze-dwell as input, the user can revert back to the SG interface or progress to the more detailed Holo-box mode.

Level 3: Holo-boxes (HB) (Fig.75 right), are 3D interfaces summoned at will or established at predetermined locations. The HB is an isolated space where the user can freely interact with virtual content using direct input without environmental occlusion or safety hazards. HB interfaces include both 2D and 3D content scaled to the size of the HB itself and to the user's available space.

5.2 Machine Shop context

Holo-box is applied in the context of a machine shop. Users wear a Magic Leap while operating milling or turning machinery accessing virtual content through the Holo-box's UI. Augmented content may vary from notifications to visualizing complex 3D objects for manufacturing. Virtual interfaces use the presented three-tier non-intrusive LOD architecture while the user's focus is on the real world. Milling and turning tasks are presented gamified as "missions" consisting of a 3D or 2D object representation to manufacture as well as a textual description of the manufacturing process [2]. By using Holo-box's architecture, missions are presented in a gamified manner as follows:

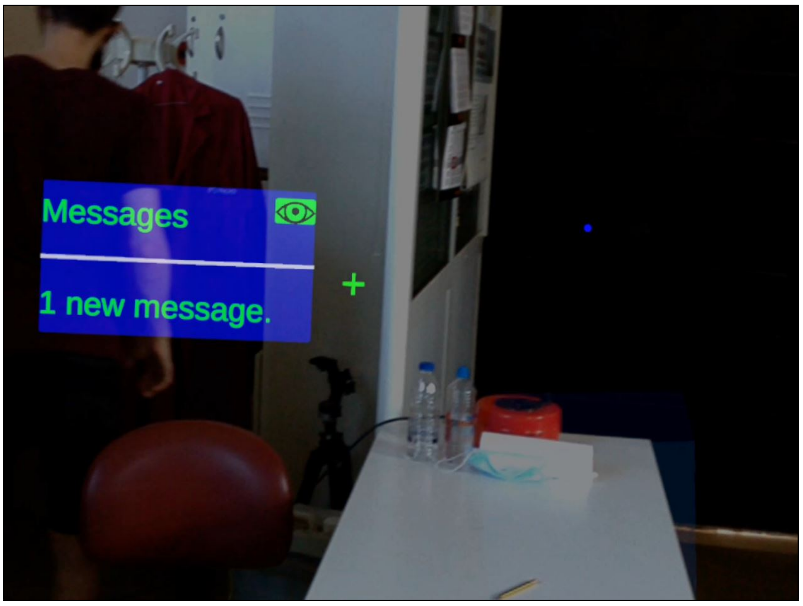


Figure 76: Simple Glanceable prototype

Level 1: SG (see Fig. 76) Notification that a new mission is available which may pop up at any time while the user is working. The "eye" button transitions to the DG interface.

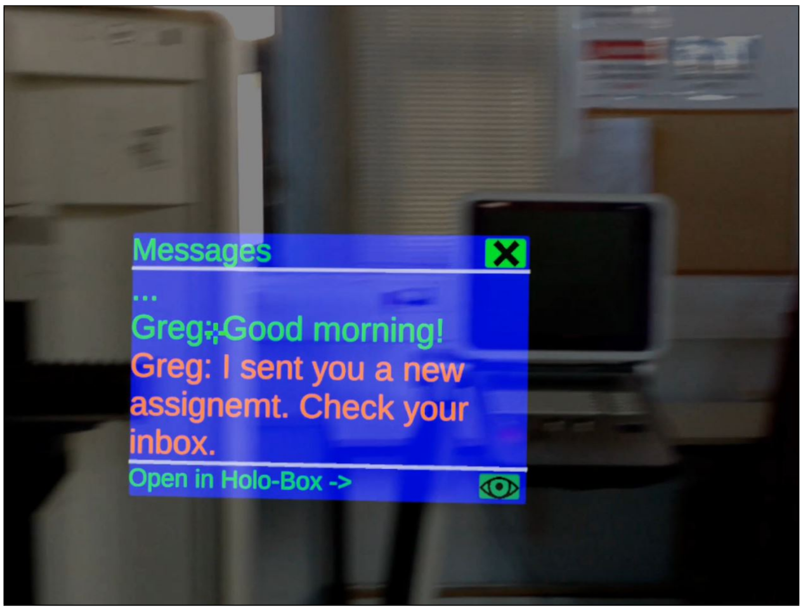


Figure 77: Detailed Glanceable prototype

Level 2: DG (see Fig. 77) Simple textual description of the mission, which is shown after pressing a button on the SG interface. The placeholder text will be replaced by a textual description of the given task. The "eye" button transitions to the HB interface. The "X" button reverts to the SG interface.

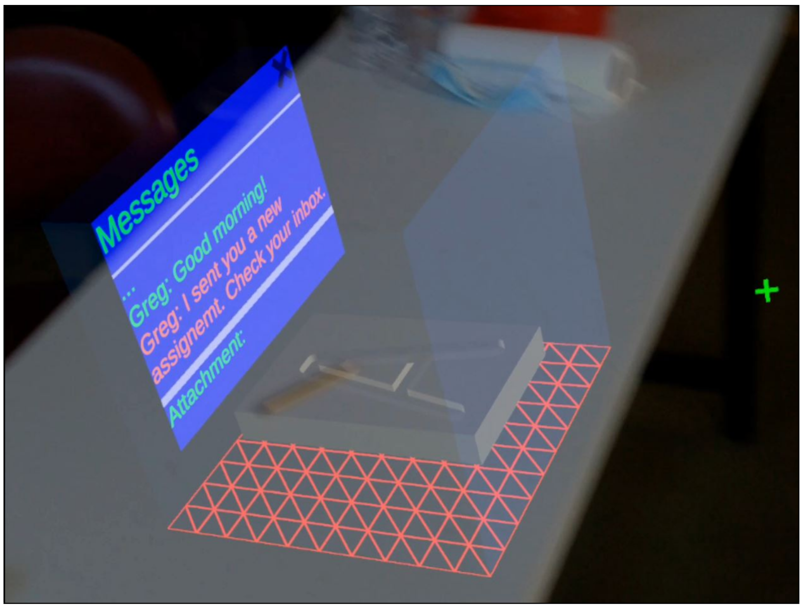


Figure 78: Holo-Box prototype

Level 3: HB (see Fig. 78) Detailed textual description of the request and an interactable 3D model of the requested object to manufacture, shown on demand after pressing a button on the DG interface at a Holo-box placed on the 3D space. The placeholder text is the same as the DG interface. By interacting with the 3D object by hovering their index finger over the 3D model the user can view additional information including measurements of the finished product as seen in the mechanical blueprint of the object (see Fig. 61).

5.3 Issues and limitations

During the implementation of this prototype we found certain issues or limitations in our proposed interface system, which we aimed to resolve during the course of our final implementation. These issues included the following:

Inaccurate eye tracking. For our implementation we utilised the Magic Leap One's on board eye tracker. The eye tracker calculates an estimated fixation point in 3D space, based on triangulating both eyes according to the Magic Leap's initial calibration data, in addition to a confidence value that grades the accuracy of the readings. In Unity, the fixation point is shown as a 3D point in space which changes values only when a new fixation point is calculated with a high confidence value, otherwise the fixation point remains stationary on its previous value. We translated this 3D point into a 2D cursor which only moved horizontally and vertically towards the fixation point and interacted with objects behind it regardless of depth. In addition, the cursor's movement was smoothed to follow the fixation point at a fixed speed in order to avoid sudden jumps and rapid flickering (as seen in Fig. 76 where the blue circle represents the fixation point and the green cross the cursor moving towards it).

In practice, the eye tracker was often unable to track the user's eye effectively resulting in either the cursor freezing in place making the user unable to interact with interfaces or the eye tracker detected false eye movements and moved onto interactable objects resulting in false interactions.

In order to solve this, we used the Magic Leap Control for two functions. First, the user could use the onboard trackpad to introduce a small offset from their fixation point by manually moving the target when the tracking failed. Second, we changed our passive gaze-delay interactions, where if the cursor remained on top of an interactable object for 1 second it would invoke an interaction, with an active interaction by pressing a controller button.

Through this change we understood that the eye tracking cursor should have some secondary form of movement to enhance its accuracy, and gaze-based interactions should be done through a form of direct interaction instead of happening passively under specific conditions.

For the final implementation, the Glanceable interfaces were unlocked from the user's field of view and instead were fixed to the user's surrounds at a set angle and a set distance. This way the user can use a combination of eye tracking and head movement to move the gaze-based cursor in relation to the Glanceable interfaces. In addition, the Magic Leap's eye tracker also detects whether the user is blinking or not with one value for each eye. Thus, we replaced the clicking of a button on the controller with the blinking of one eye, which resulted in a direct form of interaction which only uses the user's eyes.

Confusing interaction switch between interfaces. As mentioned before, users were shown how to use gaze-based interactions with a delay for gaze-based interfaces, while the Holo-box used instantaneous hand-based interactions, which proved to be disorienting.

In order to solve this, for the final implementation both interaction types were combined into a universal system. Both forms of interaction used separate cursors, one for the eye-gaze and two for each index finger and an interaction occurred when the user either started blinking with the eye-gaze cursor on top of an object or when a hand cursor entered an object's trigger zone. Both actions performed an interaction with a small delay to minimise misinteractions and hand-based interactions took priority over gaze-based interactions if both occurred simultaneously to minimise conflicting cases. As a safety measure, hand-based interactions only work inside the Holo-box interfaces and not on Glanceable.

SG LOD is underused. The initial concept for the SG interface was to show a generic message notifying there was a change inside the AR interfaces, which provided no useful information without opening at least one additional interface.

For the finished implementation this was changed to show an informative message that was limited to a few words. The message informed the user of a new manufacturing task and included the name of the product to manufacture and the material to use (e.g. "12mm cog, copper"). Additional information on the given task are received through subsequent interfaces, but this amount of information could be enough to inform an experienced user that has manufactured the same object before and memorised it.

Additional machine shop information was needed. Even with the proposed 3-LOD interface system, the information given to the user was only relevant to the dynamic data of the given task. In the previous work of G-code Machina [2] the users first received a manufacturing task and then read the relevant documentation to gain additional knowledge required to complete a mission.

In our work, users did not need to read documentation regarding the G-code, but they had to find additional objects to complete their given task, such as manufacturing tools. Thus, the Holo-box was extended to have additional content than included a visualised documentation in addition to the given mission's description. This documentation could also include 3D objects which could be directly compared to their real world equivalents.

6 Holo-Box: Multi-LOD Glanceable Interfaces for Machine Shop Guidance using Blink and Hand Interaction

In this chapter we analyze the final design of our proposed Glanceable interface system Holo box developed using Unity 2020 and the Magic Leap One AR Head-mounted Display (HMD).

The contribution of this work is threefold:

- Holo-box is a novel, gaze-activated 3-LOD system that initially launches with a compact interface and progressively expands to reveal more information with each LOD. Two initial LOD interfaces are controlled through glance (Glanceable interfaces), while the deepest level of LOD is a 3D world-based interface. The interface enables the user to view simple guidance information through the Glanceable interfaces or select to view the 3D interface to examine more detailed information.
- Holo-box utilises a new interaction system that utilizes a combination of eye-gaze blinking with a delay (dubbed blink-delay) for interacting with Glanceable interfaces to minimize unintended interactions. Objects remain selected and interactable even if the eye cursor exits their activation trigger unless another interactable object is selected. Glanceable interfaces are also visually adjusted to minimize errors in interaction, deliberately placing buttons on opposite sides of the interface. The 3D interfaces also allow hand tracking interactions with a delay (dubbed hand-delay). Users can engage interchangeably between hand-delay and blink-delay interactions; thus, interactable objects can be placed on any layout without restrictions.
- The content shown in each LOD is adjusted so that each interface can be used to guide users with different levels of knowledge on its own. In the most compact Glanceable interface, we show only textual information, including the name and material of a given task. The second LOD level of the Glanceable interface offers additional text and 2D images. The subsequent 3D interface includes 3D objects and multiple interactable buttons. The Glanceable interfaces provide adequate guidance for experienced users, while the 3D interface provides additional information for inexperienced users, using 3D models relevant, in our case, to manufacturing tasks.

We evaluate Holo-box by employing an object selection task in a pilot manufacturing scenario. The content displayed on all interfaces is used in presenting milling and turning manufacturing tasks in the form of gamified missions. Experiment results show that users complete missions faster and use more compact interfaces as they gain more experience. Our proposed interaction system compensates for the eye tracker's inaccuracies resulting in higher perceived accuracy on blink-delay inputs than hand-delay.

The implementation can be separated into three chapters: (1) The interface design, (2) User interactions, and finally, (3) Adapting the content of the interfaces to take full advantage of the designed interface system.

Our proposed design aims to achieve three primary design goals (DG) that include:

- *(DG1) The system must be usable while working on a real world task* while minimizing potential safety risks, improving on information retrieval and safety practices of previous work in AR guidance in the manufacturing workspace [68].
- *(DG2) The interface must be able to show varying amounts of information in different LODs* based on whether the user's focus is on the real world or the virtual content. The user's focus is governed by direct interactions with virtual objects instead of automatically [17], reducing potential distractions from shifting interfaces while the user's focus is on a real-world task.
- *(DG3) The interfaces should achieve accurate interactions and the Midas Touch problem should be minimized when using gaze interaction.* To mitigate the impact of the Midas touch problem, a direct form of interaction, such as blinking, is preferred[73]. In dense interfaces where interactable objects are prone to overlap [71], a different interaction paradigm should be used such as hand-tracking. Speech control is not appropriate due to the noisy nature of the manufacturing workspace.

6.1 Interface design

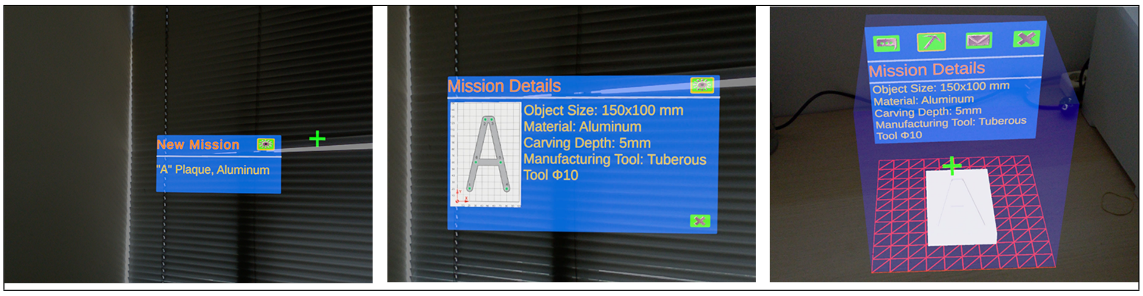


Figure 79: Holo box. 3-LOD interface design. Level 1: Simple Glanceable - SG (Left), Level 2: Detailed Glanceable - DG (Middle), Level 3: Holo-Box - HB (Right)

Holo-box employs the same 3-LODs (e.g. Simple Glanceable (SG), Detailed Glanceable (DG) and Holo-Box (HB). See Fig. 79) mentioned in the prototype design, but the usability of each interface has changed to adapt to the issues and limitations mentioned in the previous chapter as well as the Design Goals mentioned in this chapter.

SG and DG are 2D display-fixed Glanceable interfaces that follow the user's head orientation and are always visible. HB is a 3D interactive interface which remains fixed at a predefined anchor location in the real world chosen by the user. The usability of each interface is as follows:

Simple Glanceable (SG): (Figure 79 left) The SG interface is a compact 2D interface that shows minimal information limited to a few words. SG minimizes real-world occlusion and is the first interface encountered to notify the user new data is present in the virtual interfaces. The SG interfaces include an interactable button that transitions to the DG interface.

Detailed Glanceable (DG): (Figure 79 middle) The DG interface provides more detailed information to the user, including a few lines of text as well as 2D media such as images. DG is larger than the SG interface, but it is still relatively small compared to the Magic Leap's FoV to not impede a large area of the available FOV. DG includes two interactable buttons, one to return to the SG and another that transitions to the HB. Locating the two buttons far from each other minimizes potential unintentional interactions.

Holo-Box (HB): (Figure 79 right) The HB interface is a 3D interface that appears in the real world. During the application setup process, the user manually places the HB on the real world, designated as a semi-transparent bounding box. Its position persists during the execution of the application or until the user manually repositions it. Upon HB activation through the DG interface, the content appears inside the predetermined position regardless of the user's position during the activation. The content of the HB includes complex 2D interfaces, 3D objects and any number of interactable objects. HB also allows for both blink-delay and hand-delay interactions; these can activate interchangeably.

The two Glanceable interfaces allow for rapid information retrieval "at a glance" with two levels of information. SG is compact, allowing for easier focus on the real world providing guidance, while DG triggers after the user interacts with the SG interface. Thus, we can assume the user wants to focus on the virtual content when the DG interface appears. SG and DG interfaces follow the user at a set distance and viewing angle as they move around the real world. On the other hand, HB is anchored on a specific real-world position and does not move. HB content is presented only on the inside of its bounding box. We assume that the user has placed it appropriately at a position where they can use hand tracking without risk of injury. This design fulfills DG1 and DG2.

6.2 User Interactions

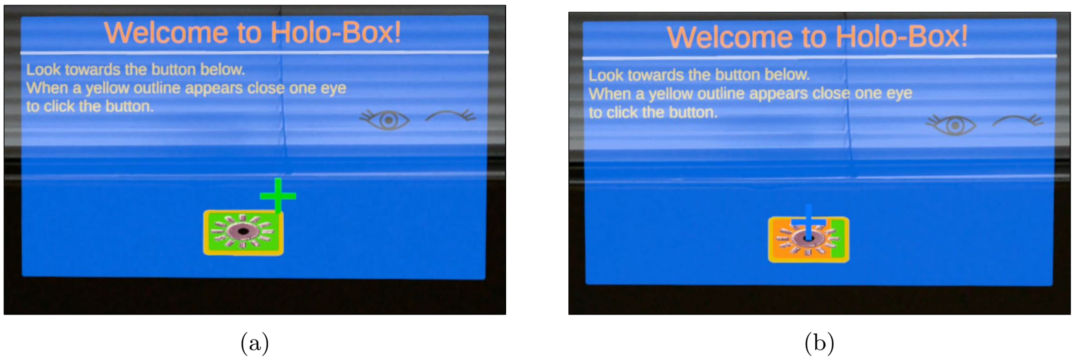

Welcome to Holo-Box!
Look towards the button below.
When a yellow outline appears close one eye
to click the button.
(a)
(b)

Figure 80: Eye tracking interaction. When one eye is closed the cursor turns blue and initiates the click.

Holo Box allows for two types of interaction, one that uses eye tracking (see Fig. 80) and one that uses hand tracking (see Fig. 81). Both interaction types are designed around using delay-based inputs, where the user targets an object using a cursor and remains on the target for a set amount of time before the interaction happens. To reduce gaze mis-interactions, the user must place the eye-tracking cursor and then blink with one eye to interact, thus, using "Blink-Delay" interactions.Hand tracking interactions on the other hand use different cursors which are placed on the users' index fingers on both hands.The user interacts with objects by positioning their fingers in a trigger zone which extends a few centimeters around the interactable object and remain inside the trigger for the same delay period for a "Hand-Delay" input. To avoid conflicting simultaneous interactions, hand tracking is prioritized over eye-tracking interaction. If both hands engage simultaneously, then the left-hand cursor is prioritized.

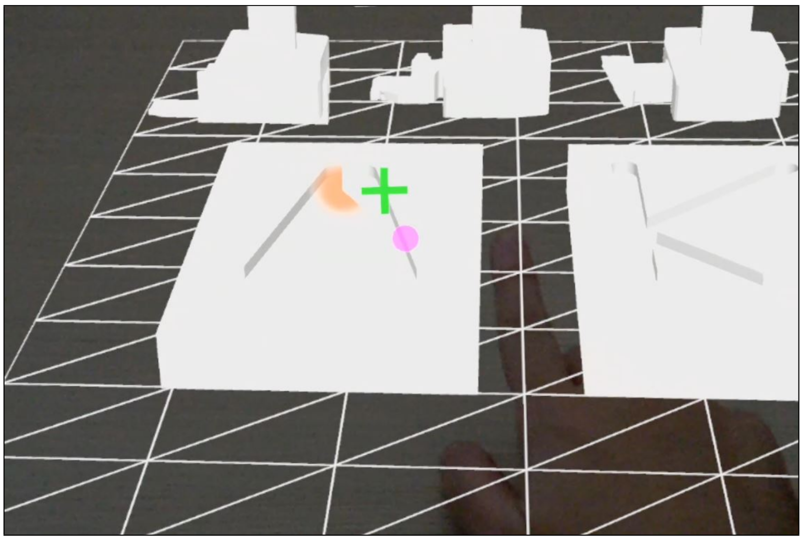


Figure 81: Right hand cursor (purple circle). The user's hand appears to the right of the cursor due to the Magic Leap camera's recording perspective.

Interactable objects in Holo-box operate using a universal state machine. An interactable object can be at one of three states: Idle, Highlighted or Clicked (see Fig. 82). Initially, all objects are in an idle state, showing their basic visuals. When a cursor enters the object's trigger zone, that object becomes highlighted, enabling an additional visual effect to show the transition. After a click is initiated by blinking on a Blink-Delayed input or immediately when highlighted on Hand-Delay inputs, a progressively filling visual effect is presented, showing the progress towards the interaction completion. After the delay period has passed, the click effect is triggered and the object returns to an idle state.

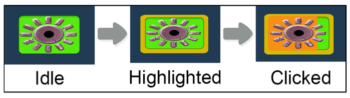


Figure 82: Button States

In Holo-box, to fulfill DG3, only one object can be at the highlighted and clicked states at all times shared across all interaction types. The highlighted object is deselected only after a click, or another object becomes highlighted and not when the user exits the trigger zone. When using Hand-Delay, the object changes from clicked to highlighted on exit. This way the user can select an object and even if their cursor moves outside the trigger zone due to tracking errors, the delayed click can still be performed by blinking if the cursor does not select a different object.

To fulfill design goal DG1, Hand-Delay interactions are disabled on the SG and DG interfaces, as these interfaces move and may be placed in a dangerous location for hand motions. Based on the design,the SG and DG interfaces are designed to minimize misinteractions. SG only comprises a single interactable button. DG includes two buttons placed on the top and bottom edges of the interface to reduce misinteractions when using Blink-Delay inputs. HB allows for both Hand-Delay and Blink-Delay inputs. As such, there are no restrictions on the amount of interactable objects as long as they are placed so that their trigger zones are not overlapping.

6.3 Machine shop Content Adaptation

A set of Glanceable interfaces were designed for this work, including information guiding the user to perform the experimental procedure. In the experiment, the users are given a set of instructions and they are tasked with selecting the correct object corresponding to a manufacturing task, including a **cutting tool**, a **cutting material** and a **manufactured product**.

For the Glanceable interfaces, each manufacturing task is presented in the form of a gamified ”mission” [2]. A mission consists of a set of information required to perform a manufacturing task setup. This consists of which tool and material to use, a 2D blueprint of the finished product and numerical cutting parameters such as the cutting speed. Each mission has a predefined tool, finished product and manufacturing parameters, but the material can vary.

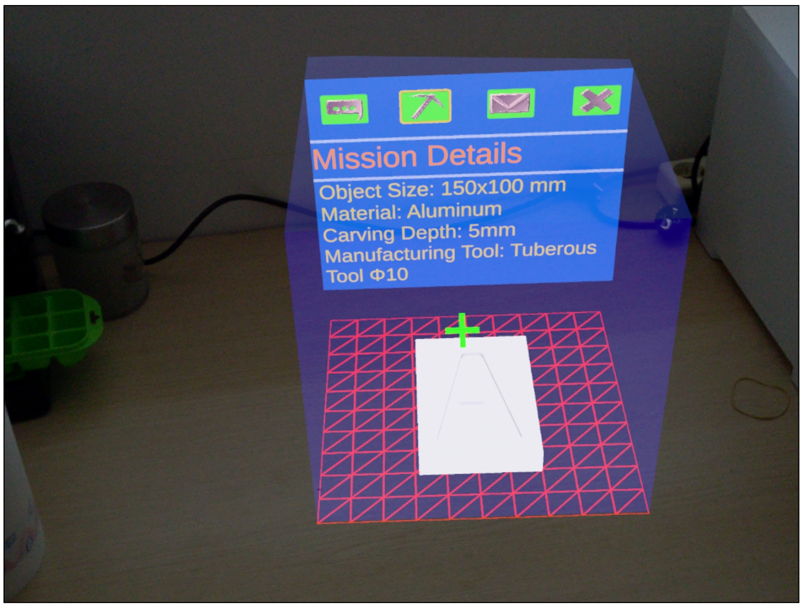
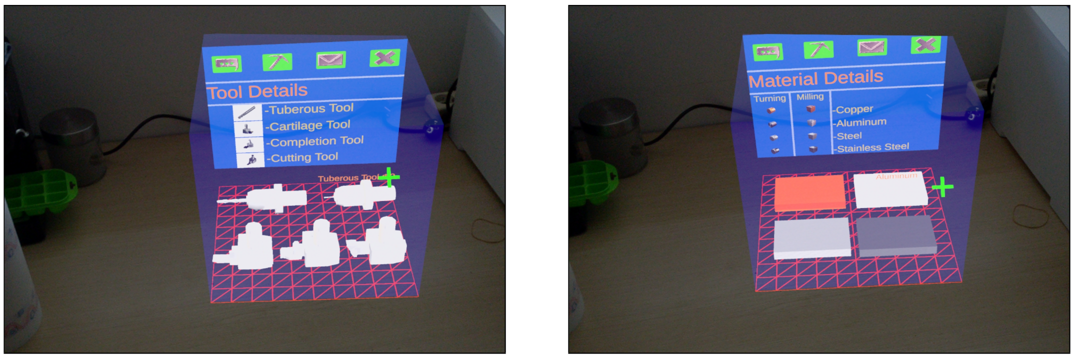


Figure 83: The Holo-box interface mission description.

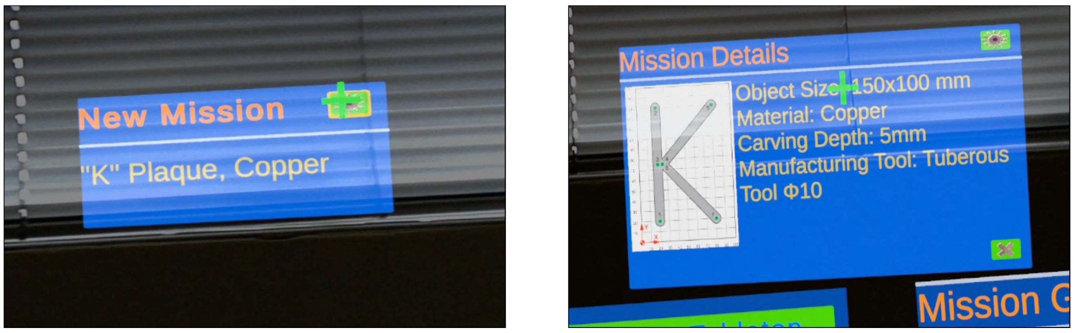
The presentation of the missions is adapted to allow efficient guidance with any of the three interfaces according to the user's expertise. The HB interface contains the information required to complete any mission. This includes mission information (see Fig. 83) as well as tool (see Fig. 84a) and material (see Fig. 84b) look-up tables with added 3D objects for better visualisation. The 3D models of the tools and materials use a modified version of the interaction logic that shows the object's name above the model when highlighted.



(a) (b)

Figure 84: Holo-Box interfaces. (a) Tools database, (b) Materials Database

The DG interface includes mission-specific information in a 2D interface (see Fig. 85b). The SG interface only contains the name of the mission as well as the correct material (see Fig. 85a). The SG interface is adequate for experienced users who have already completed a given mission and memorized it. To make the SG interface more efficient, each mission was given a different name, e.g., the "A plaque" corresponds to cutting the letter "A" on a square block.



(a) (b)

Figure 85: Simple Glanceable and Detailed Glanceable interfaces

6.4 Evaluation interface

The Evaluation Interface (EI) used in the evaluation is a modified version of the HB with the same functionality. This interface includes 2D interfaces with buttons used to operate the evaluation procedure, e.g., starting the evaluation or moving to the next mission providing feedback to the user (Figure 86). In addition, the interface includes 3D models of all available machining tools, materials and finished products which the user can select by interacting with them to complete a given training mission.

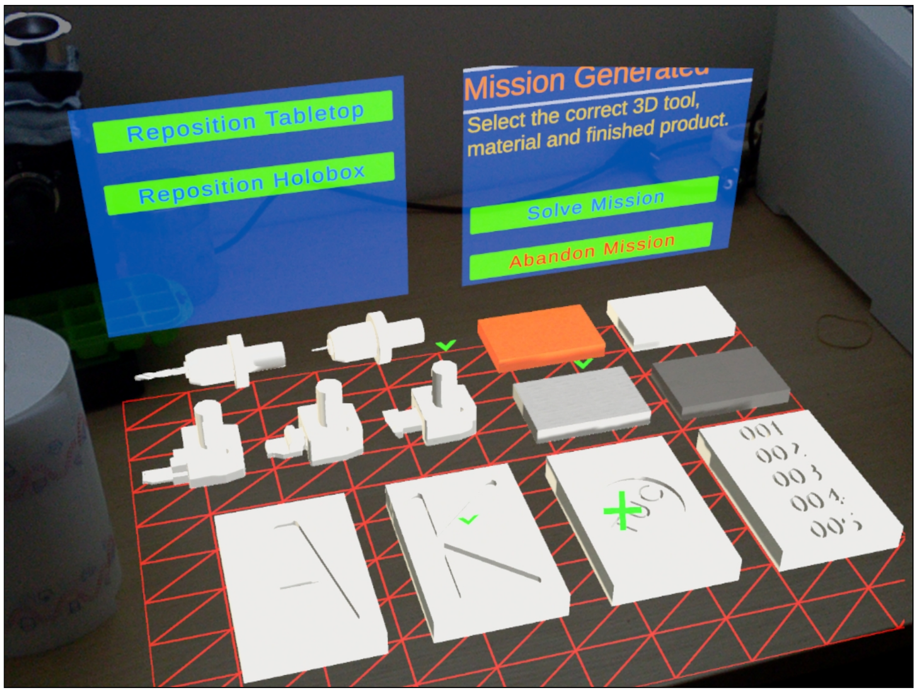


Figure 86: Evaluation Interface

7 Low level Unity Implementation

In this chapter we will analyze the implementation of our work at a low level. Our implementation was done exclusively through Unity, which was linked with Magic Leap's "The Lab" application for zero iteration debugging and a single click build and install.

7.1 Scene Hierarchy

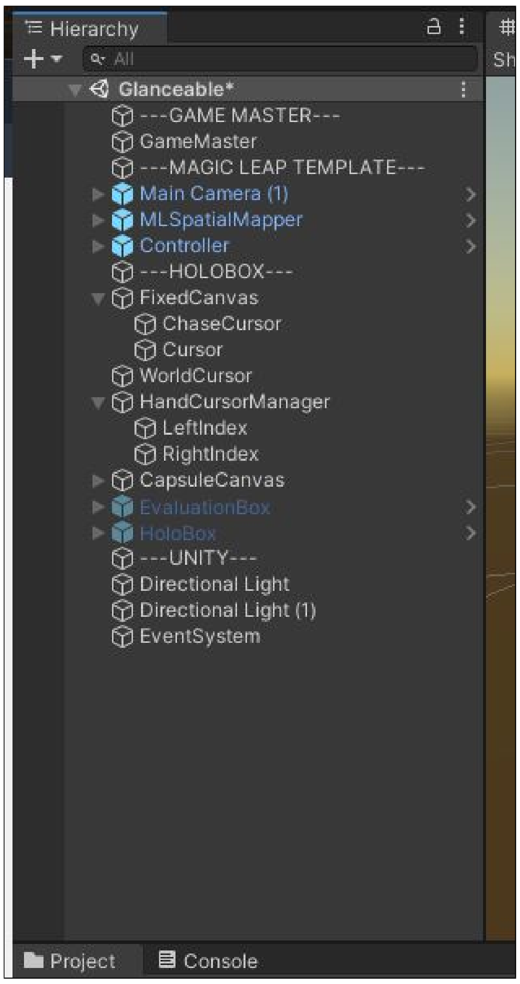
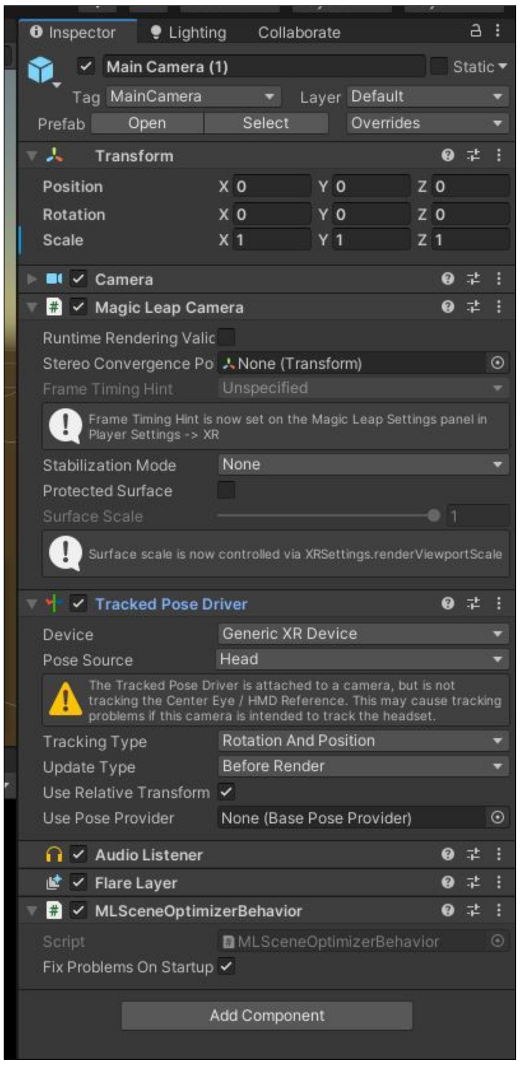


Figure 87: The implementation's scene hierarchy

Our implementation was done in a single virtual scene, dubbed Glanceable (see Fig. 87) as an internal name. The game objects inside our scene can be categorised in four categories:

- Magic Leap Template objects. These are standard pre-made object prefabs included in the Magic Leap Unity Template, which link our project with the Magic Leap One hardware appropriately.
- Unity standard objects. These are standard Unity objects present in any scene.
- The Game master. This game object holds the core logic of our work including global parameters and calls from and to other scripts.
- Holo-Box's game objects. These objects and their logic were fully implemented during this work and represent actual virtual objects inside the scene.

7.1.1 Magic Leap Template objects



Inspector | Lighting | Collaborate

✓ Main Camera (1) | Static

Tag: MainCamera | Layer: Default

Prefab: Open | Select | Overrides

Transform

	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

✓ **Camera**

✓ **Magic Leap Camera**

Runtime Rendering Valic	☐
Stereo Convergence Po	None (Transform)
Frame Timing Hint	Unspecified

Frame Timing Hint is now set on the Magic Leap Settings panel in Player Settings -> XR

Stabilization Mode	None
Protected Surface	☐
Surface Scale	1

Surface scale is now controlled via XRSettings.renderViewportScale

✓ **Tracked Pose Driver**

Device	Generic XR Device
Pose Source	Head

The Tracked Pose Driver is attached to a camera, but is not tracking the Center Eye / HMD Reference. This may cause tracking problems if this camera is intended to track the headset.

Tracking Type	Rotation And Position
Update Type	Before Render
Use Relative Transform	✓
Use Pose Provider	None (Base Pose Provider)

✓ **Audio Listener**

✓ **Flare Layer**

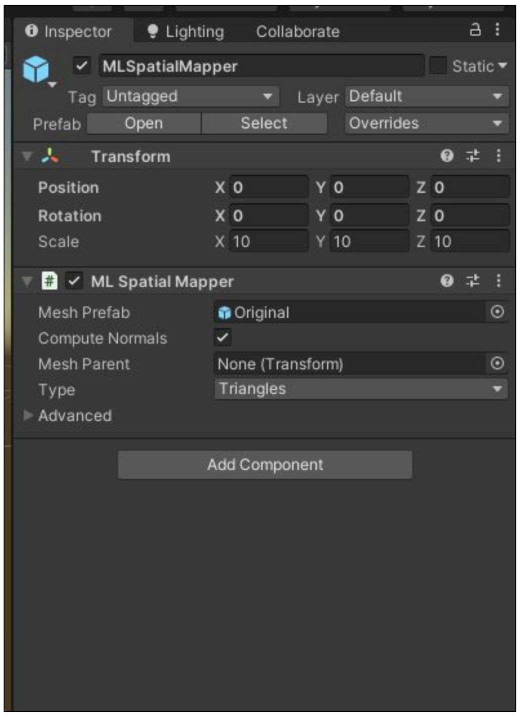
✓ **MLSceneOptimizerBehavior**

Script	MLSceneOptimizerBehavior
Fix Problems On Startup	✓

Add Component

Figure 88: The Main Camera Object's inspector

Main Camera (see Fig. 88). The Main camera represents the user's eyes into the virtual scene. Through the *Magic Leap Camera*, *MLSceneOptimizerBehavior* and *Tracked Pose Driver* components the game object tracks the physical hardware and adapts virtual content as the user moves around the virtual scene to keep the real and virtual worlds aligned at all times.


Inspector | Lighting | Collaborate

✓ MLSpatialMapper — Static

Tag: Untagged — Layer: Default

Prefab: Open | Select | Overrides

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 10	Y 10	Z 10

✓ **ML Spatial Mapper**

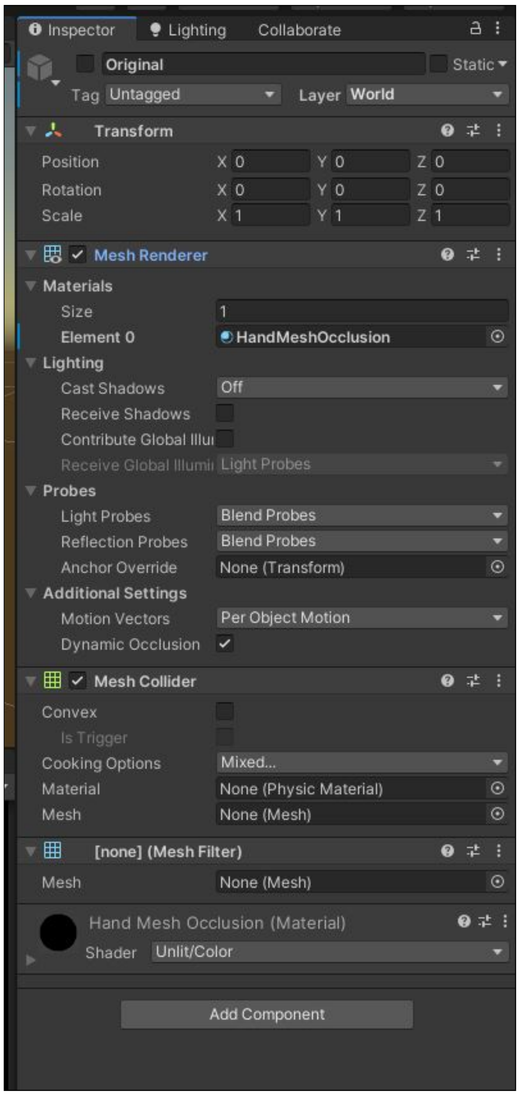
Mesh Prefab	Original
Compute Normals	✓
Mesh Parent	None (Transform)
Type	Triangles

Advanced

Add Component

Figure 89: The ML Spatial Mapper Object's inspector

ML Spatial Mapper (see Fig. 89). This object is responsible for scanning the mesh of the real world and then creates a collection of polygonal meshes inside the virtual scene. These meshes allow us to visualise the scanned real world's shape and add occlusion and collision to these objects if necessary. The visuals and behavior of the scanned mesh objects are based on the *Mesh Prefab* field which specifies the game object to use as a template.


Inspector | Lighting | Collaborate

Original — Static
Tag: Untagged — Layer: World

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Mesh Renderer ✓

Materials	
Size	1
Element 0	HandMeshOcclusion
Lighting	
Cast Shadows	Off
Receive Shadows	
Contribute Global Illu	
Receive Global Illumi	Light Probes
Probes	
Light Probes	Blend Probes
Reflection Probes	Blend Probes
Anchor Override	None (Transform)
Additional Settings	
Motion Vectors	Per Object Motion
Dynamic Occlusion	✓

Mesh Collider ✓

Convex	
Is Trigger	
Cooking Options	Mixed...
Material	None (Physic Material)
Mesh	None (Mesh)

[none] (Mesh Filter)

Mesh	None (Mesh)

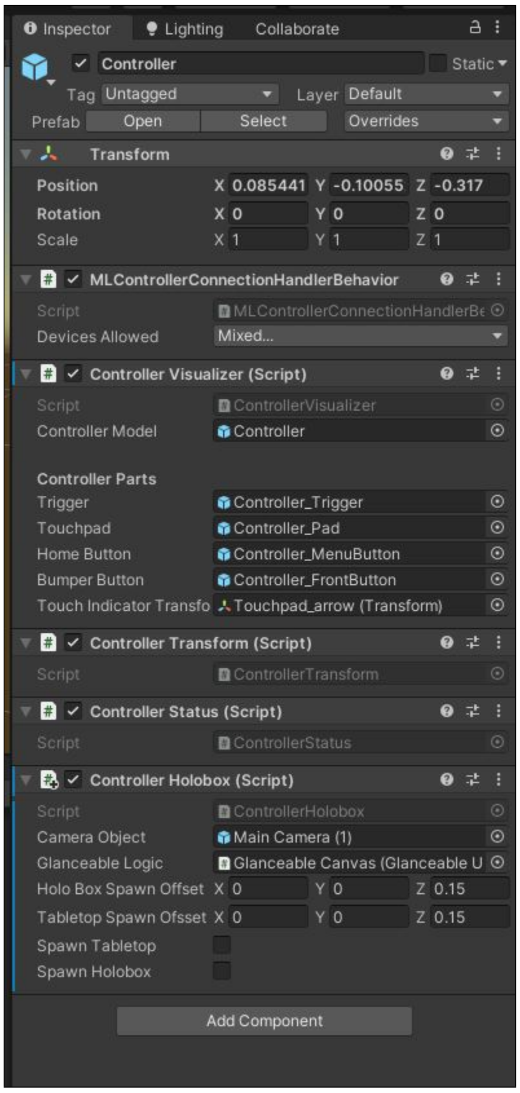
Hand Mesh Occlusion (Material)
Shader: Unlit/Color

Add Component

Figure 90: The ML Spatial Mapper's Mesh Template Object's inspector

The Mesh template object is provided by default along the Spatial mapper (see Fig. 90). By default, the mesh's objects are visualised as a multi-colored mesh visible to the users. In our case, we switched the default *Wireframe* material with the *HandMeshOcclusion* one, which has no visuals on its own, but occludes other objects behind it. This way, if a virtual interface intersects with a real object it will be partially occluded.

The object's mesh collider also allows our objects to be included in physics calculations, which in our case is necessary for the eye tracking, as the eye tracker calculates the user's fixation point by raycasting from the user's eyes towards the virtual world until it collides with a virtual object.


Inspector | Lighting | Collaborate

Controller — Static

Tag	Untagged
Layer	Default
Prefab	Open / Select / Overrides

Transform

	X	Y	Z
Position	0.085441	-0.10055	-0.317
Rotation	0	0	0
Scale	1	1	1

MLControllerConnectionHandlerBehavior

Script	MLControllerConnectionHandlerBe
Devices Allowed	Mixed...

Controller Visualizer (Script)

Script	ControllerVisualizer
Controller Model	Controller

Controller Parts

Trigger	Controller_Trigger
Touchpad	Controller_Pad
Home Button	Controller_MenuButton
Bumper Button	Controller_FrontButton
Touch Indicator Transfo	Touchpad_arrow (Transform)

Controller Transform (Script)

Script	ControllerTransform

Controller Status (Script)

Script	ControllerStatus

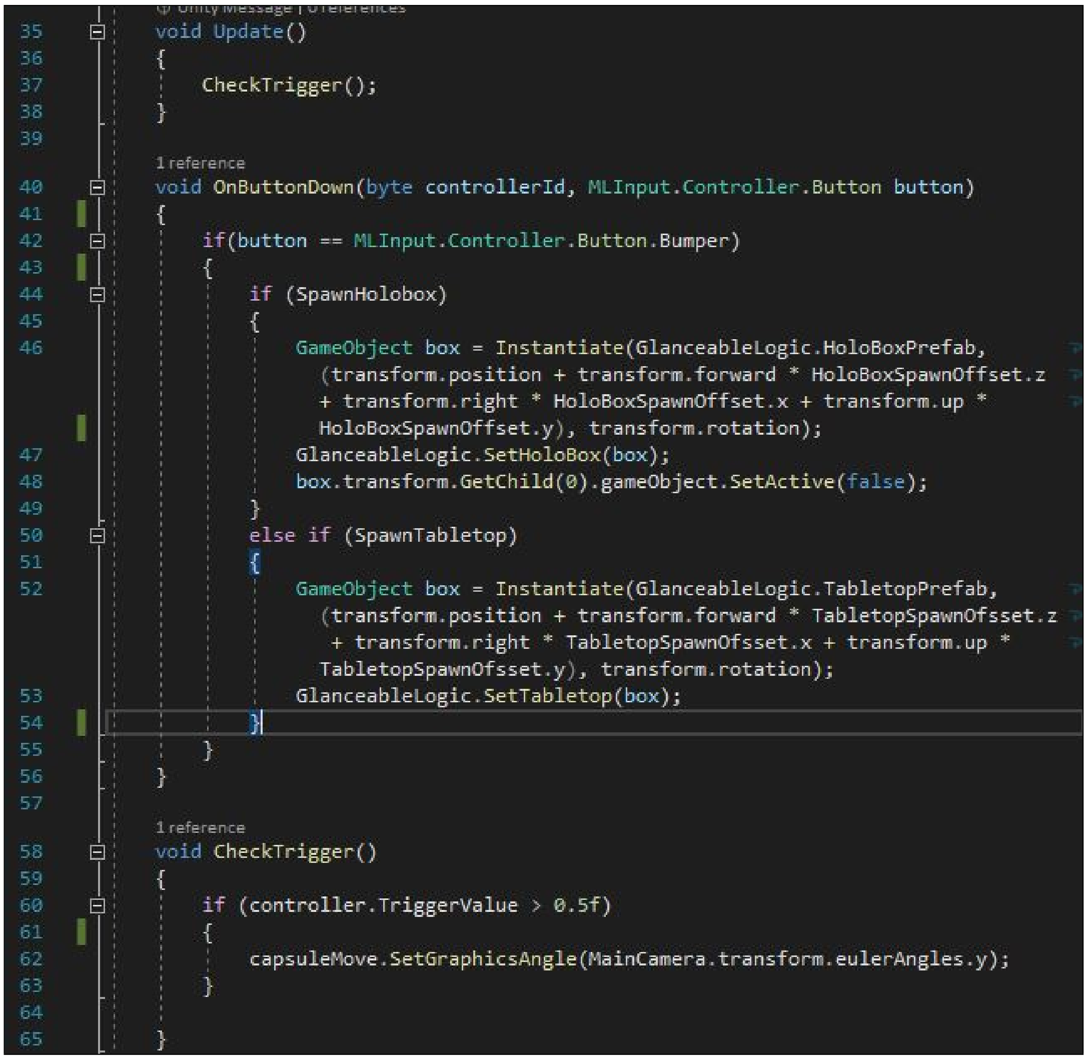
Controller Holobox (Script)

Script	ControllerHolobox		
Camera Object	Main Camera (1)		
Glanceable Logic	Glanceable Canvas (Glanceable U		
Holo Box Spawn Offset	X 0	Y 0	Z 0.15
Tabletop Spawn Offset	X 0	Y 0	Z 0.15
Spawn Tabletop	☐		
Spawn Holobox	☐		

Add Component

Figure 91: The Controller Object's inspector

Controller (see Fig. 91). The Controller is responsible for both tracking and visualising the Magic Leap Control. The *Controller Visualiser* script is responsible for visualising the controller as a virtual object inside the scene. The *Controller Transform*, *Controller Status* and *MLController-ConnectionHandlerBehavior* are responsible for tracking the controller's position. The *Controller Holobox* script allows us to read and use controller inputs in our application.



```
    Unity Message | 0 references
    void Update()
    {
        CheckTrigger();
    }

    1 reference
    void OnButtonDown(byte controllerId, MLInput.Controller.Button button)
    {
        if(button == MLInput.Controller.Button.Bumper)
        {
            if (SpawnHolobox)
            {
                GameObject box = Instantiate(GlanceableLogic.HoloBoxPrefab,
                  (transform.position + transform.forward * HoloBoxSpawnOffset.z
                  + transform.right * HoloBoxSpawnOffset.x + transform.up *
                  HoloBoxSpawnOffset.y), transform.rotation);
                GlanceableLogic.SetHoloBox(box);
                box.transform.GetChild(0).gameObject.SetActive(false);
            }
            else if (SpawnTabletop)
            {
                GameObject box = Instantiate(GlanceableLogic.TabletopPrefab,
                  (transform.position + transform.forward * TabletopSpawnOfsset.z
                   + transform.right * TabletopSpawnOfsset.x + transform.up *
                  TabletopSpawnOfsset.y), transform.rotation);
                GlanceableLogic.SetTabletop(box);
            }
        }
    }

    1 reference
    void CheckTrigger()
    {
        if (controller.TriggerValue > 0.5f)
        {
            capsuleMove.SetGraphicsAngle(MainCamera.transform.eulerAngles.y);
        }

    }
```

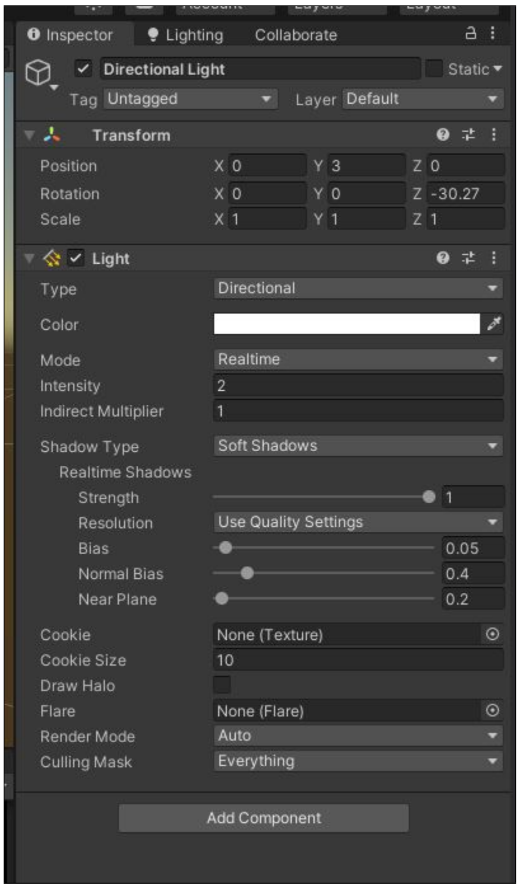
Figure 92: The Controller Holobox script

The *Controller Holobox* script contains two key functions. The *Check Trigger* function is called on every frame and checks how far the trigger is pressed (0 = not pressed, 1 = fully pressed). If more than half the trigger is pressed, we rotate the *Capsule Canvas* which holds the Glanceable interfaces to match the user's forward angle. This way, if the user presses the trigger and then rotates their head they can reposition the Glanceable interfaces to their desired angle. The button and trigger values are received through the *MLController.Controller* class included in the *UnityEngine.XR.MagicLeap* package.

The *On Button Down* function is a callback method which is automatically executed by Unity every time a button press is detected from any input (including keyboard, mouse and all types of controllers when available). In our case, we only detect if the Magic Leap Control's bumper was pressed. The only times when we want to use this button are when we want to spawn the Evaluation Interface and the Holo-box, indicated by the *SpawnHolobox* and *SpawnTabletop* booleans.

In both cases, we Instantiate a new Game Object either of type *HoloBoxPrefab* or *Tabletop-Prefab*, both of which are initialized inside the *GlanceableLogic* script, which is a class of type *GlanceableUILogic* stored onto our *GlanceableCanvas* Game Object. The object is instantiated with the same position and rotation as the Magic Leap Control. Finally, the Glanceable logic's reference to the newly spawned object is updated, deleting a previous instance of the same interface if it existed previously and keeping the last one.

7.1.2 Unity Standard objects


Inspector | Lighting | Collaborate

Directional Light — Static

Tag: Untagged — Layer: Default

Transform

Position: X 0, Y 3, Z 0
Rotation: X 0, Y 0, Z -30.27
Scale: X 1, Y 1, Z 1

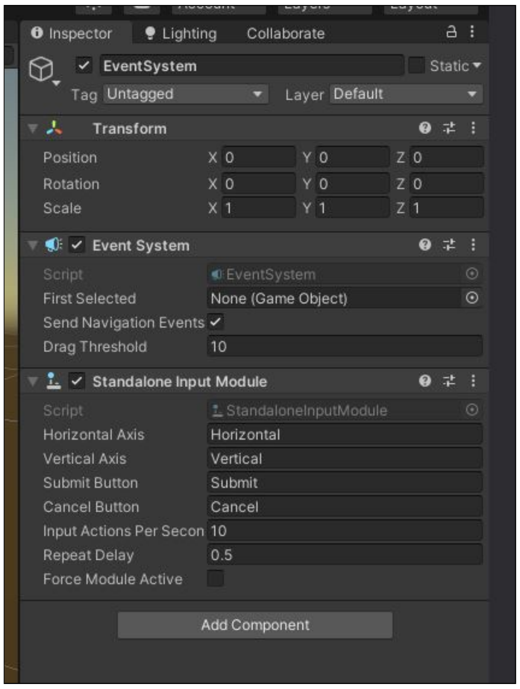
Light

Type: Directional
Color
Mode: Realtime
Intensity: 2
Indirect Multiplier: 1
Shadow Type: Soft Shadows
Realtime Shadows
Strength: 1
Resolution: Use Quality Settings
Bias: 0.05
Normal Bias: 0.4
Near Plane: 0.2
Cookie: None (Texture)
Cookie Size: 10
Draw Halo
Flare: None (Flare)
Render Mode: Auto
Culling Mask: Everything

Add Component

Figure 93: The Directional Light Object's inspector

Directional Light (see Fig. 93). The Directional light object provides a universal ambient lighting to all objects in the scene. This light simulates natural sunlight that lights every object in the scene from a set angle (identified by its rotation). In a standard 3D scene, this light would cast shadows after it collides with 3D objects and indirect light bounces would light up occluded areas with smaller intensity like the real world. In AR however, this process does not work as intended, as Magic Leap denotes lower intensity as object transparency, resulting in objects appearing fully visible on one side and transparent on the other side. To solve this, we added two Directional lights with different angles of rotation (X rot = 0^o for the first and X rot = 180^o for the second) and different intensity values (2 for the first and 1 for the second).



EventSystem Static
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

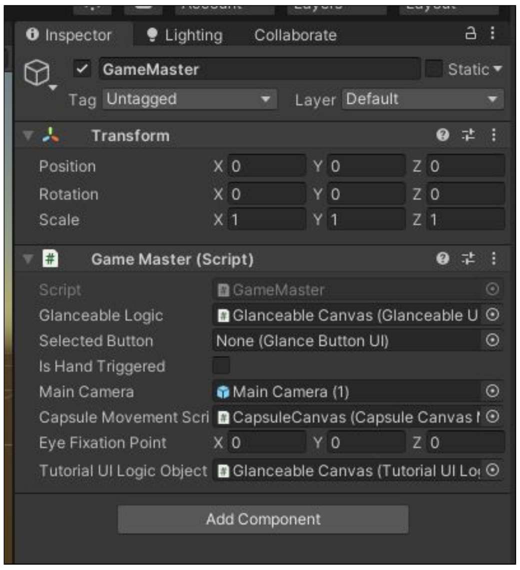
Event System
Script EventSystem
First Selected None (Game Object)
Send Navigation Events ✓
Drag Threshold 10

Standalone Input Module
Script StandaloneInputModule
Horizontal Axis Horizontal
Vertical Axis Vertical
Submit Button Submit
Cancel Button Cancel
Input Actions Per Secon 10
Repeat Delay 0.5
Force Module Active

Figure 94: The Event System Object's inspector

Event System (see Fig. 94). Unity's event system is responsible for reading inputs and calling the appropriate functions in all available scripts such as the *OnButtonDown* function inside the *Controller Holobox* script as well as calling the *OnClick* functions on every Game Object with the *Button* component, which is used by all interactable objects inside Holo-box.

7.1.3 Game Master



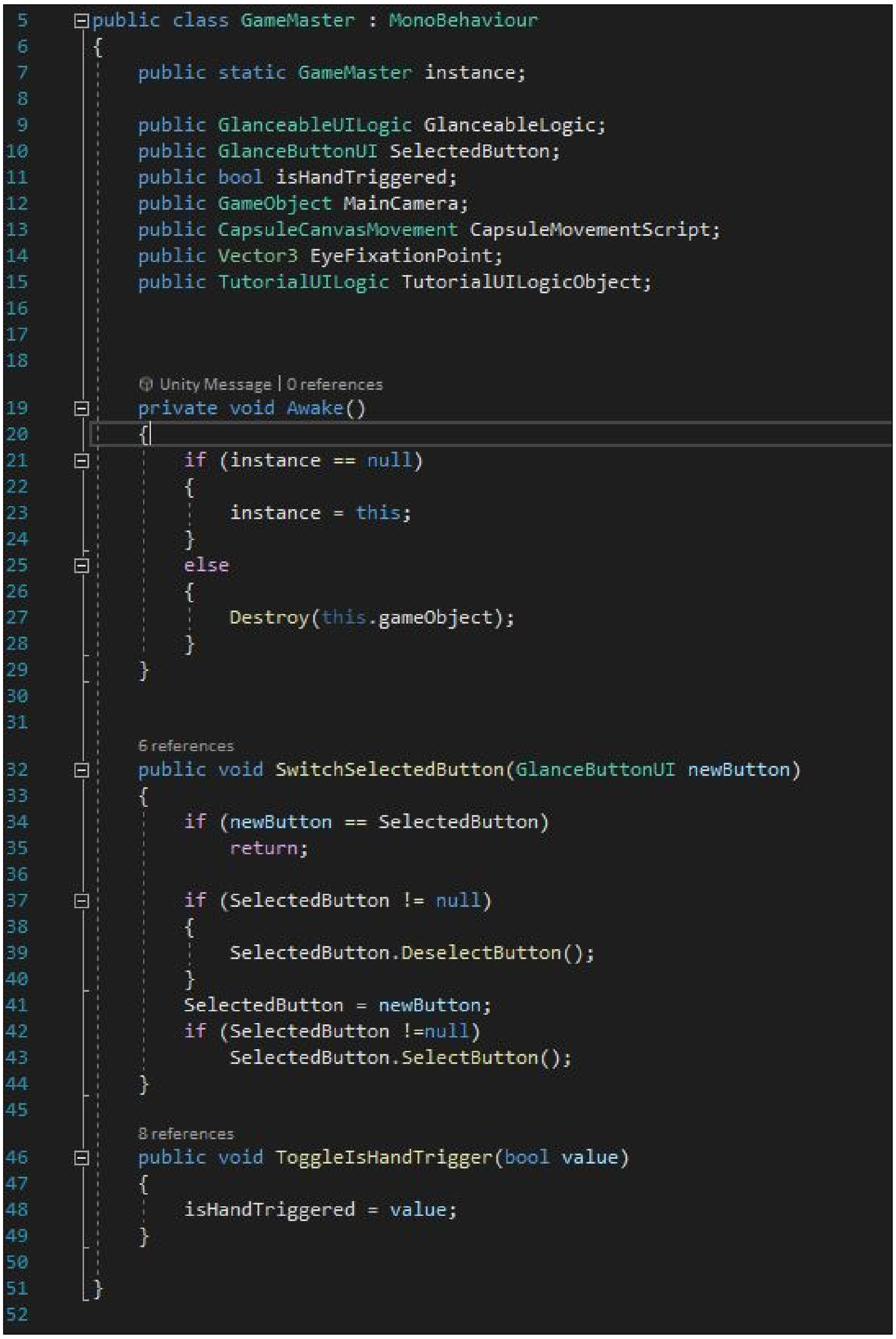
GameMaster Static
Tag Untagged Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Game Master (Script)
Script GameMaster
Glanceable Logic Glanceable Canvas (Glanceable U
Selected Button None (Glance Button UI)
Is Hand Triggered
Main Camera Main Camera (1)
Capsule Movement Scri CapsuleCanvas (Capsule Canvas I
Eye Fixation Point X 0 Y 0 Z 0
Tutorial UI Logic Object Glanceable Canvas (Tutorial UI Lo

Figure 95: The Game Master Object's inspector

The Game master is an empty Game Object with no visual or functional components (see Fig. 95) that holds the *Game Master* script.



```
public class GameMaster : MonoBehaviour
{
    public static GameMaster instance;

    public GlanceableUILogic GlanceableLogic;
    public GlanceButtonUI SelectedButton;
    public bool isHandTriggered;
    public GameObject MainCamera;
    public CapsuleCanvasMovement CapsuleMovementScript;
    public Vector3 EyeFixationPoint;
    public TutorialUILogic TutorialUILogicObject;



    Unity Message | 0 references
    private void Awake()
    {
        if (instance == null)
        {
            instance = this;
        }
        else
        {
            Destroy(this.gameObject);
        }
    }


    6 references
    public void SwitchSelectedButton(GlanceButtonUI newButton)
    {
        if (newButton == SelectedButton)
            return;

        if (SelectedButton != null)
        {
            SelectedButton.DeselectButton();
        }
        SelectedButton = newButton;
        if (SelectedButton !=null)
            SelectedButton.SelectButton();
    }

    8 references
    public void ToggleIsHandTrigger(bool value)
    {
        isHandTriggered = value;
    }

}
```

Figure 96: The Game Master script

The Game master script holds a static reference to itself, making sure *On Awake* that this script's instance is unique. Any other script can access its public parameters through this instance using the *GameMaster.instance.parameter* or *GameMaster.instance.function()* format.

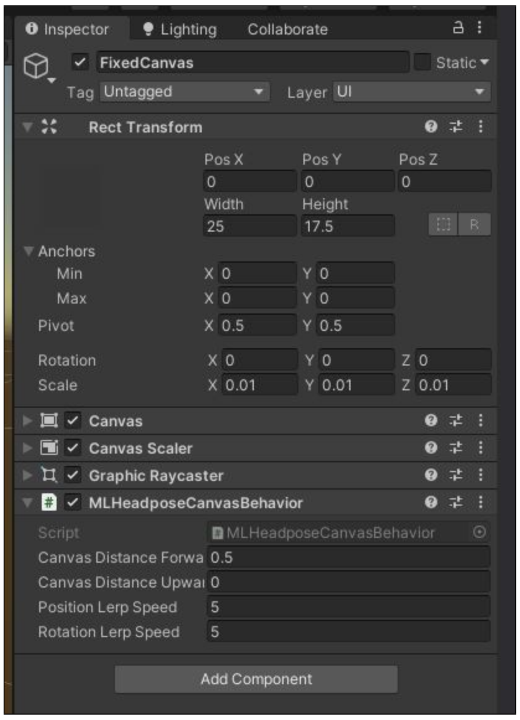
The Game master serves two functions. First, it holds public references to other objects and scripts which need to be accessed by other scripts easily as well as globally available values (*EyeFixationPoint*). Second, it manages the object selection logic, making sure only one object is selected at any given moment, stored in the *SelectedButton* parameter, and if the selected object was selected through the hand or eye cursor, stored in the *isHandTriggered* boolean parameter.

7.1.4 Holo-Box Game Objects

The Game objects we designed for Holo-box outside the Game Master fall into one of three categories:

- Cursors and their movement managers
- Interfaces with their Interface Logic
- Interface components

In the following chapters we will analyze the most important objects and scripts used by these objects.



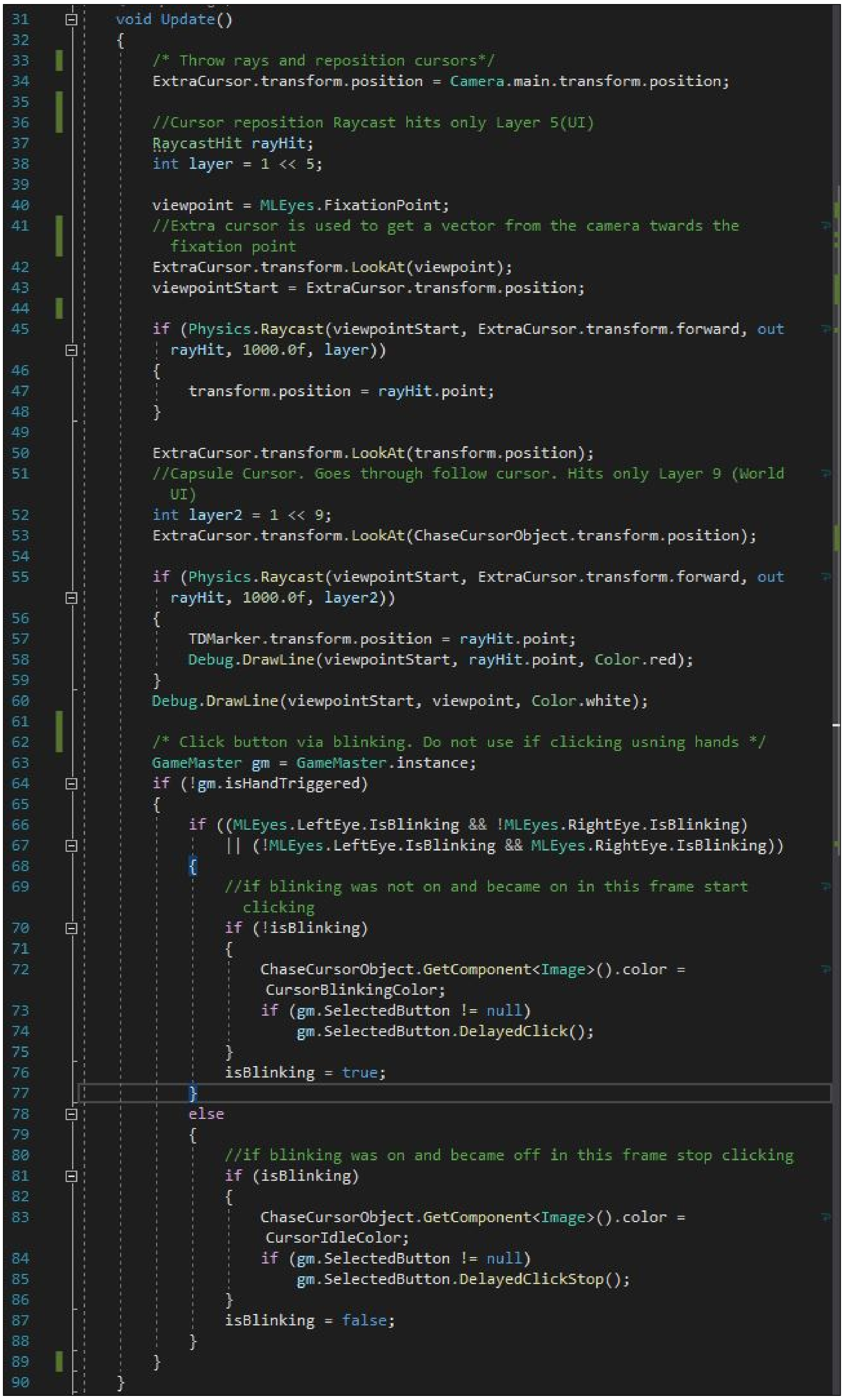
```
Inspector   Lighting   Collaborate
✓ FixedCanvas                         Static
Tag Untagged            Layer UI
Rect Transform
            Pos X      Pos Y      Pos Z
            0          0          0
            Width      Height
            25         17.5
Anchors
  Min       X 0        Y 0
  Max       X 0        Y 0
Pivot       X 0.5      Y 0.5
Rotation    X 0        Y 0        Z 0
Scale       X 0.01     Y 0.01     Z 0.01
✓ Canvas
✓ Canvas Scaler
✓ Graphic Raycaster
✓ MLHeadposeCanvasBehavior
Script                      MLHeadposeCanvasBehavior
Canvas Distance Forwa       0.5
Canvas Distance Upwa        0
Position Lerp Speed         5
Rotation Lerp Speed         5
Add Component
```

Figure 97: The Fixed Canvas Object's inspector

Fixed Canvas (see Fig. 97). The fixed canvas is a 2D panel that follows the user's viewpoint at a fixed distance using the *MLHeadposeCanvasBehavior* script provided by Magic leap. This script provides a smooth movement and the canvas's distance from the user can be adjusted using the *Canvas Distance Forward* parameter. As the Magic leap has a minimum draw distance of 0.37 meters in front of the user, we set this canvas's distance at a distance of 0.5 meters.

In the prototype implementation, this canvas held the Glanceable interfaces in addition to the eye tracking cursors, but in the final implementation it only holds the eye tracking cursors, which should always be visible in front of the user.

The fixed canvas has two children Game Objects (see Fig. 87) including the *Cursor* and the *ChaseCursor* in addition to the *WorldCursor* which is used for the same logic. The *Cursor* game object is a visual indicator that shows the realtime positon of the user's eye tracking position onto the Fixed Canvas. The *ChaseCursor* is a green cross cursor that follows the *Cursor* in a smooth motion along the Fixed Canvas. The *WorldCursor* is an invisible cursor that goes behind the *ChaseCursor* and collides with 3D objects behind it, activating the triggers of interactable objects selecting them. The *Cursor* and *ChaseCursor* objects are purely visual indicators while the *WorldCursor* has a Spherical collider with a radius of 1 millimeter.



```csharp
void Update()
{
    /* Throw rays and reposition cursors*/
    ExtraCursor.transform.position = Camera.main.transform.position;

    //Cursor reposition Raycast hits only Layer 5(UI)
    RaycastHit rayHit;
    int layer = 1 << 5;

    viewpoint = MLEyes.FixationPoint;
    //Extra cursor is used to get a vector from the camera twards the fixation point
    ExtraCursor.transform.LookAt(viewpoint);
    viewpointStart = ExtraCursor.transform.position;

    if (Physics.Raycast(viewpointStart, ExtraCursor.transform.forward, out rayHit, 1000.0f, layer))
    {
        transform.position = rayHit.point;
    }

    ExtraCursor.transform.LookAt(transform.position);
    //Capsule Cursor. Goes through follow cursor. Hits only Layer 9 (World UI)
    int layer2 = 1 << 9;
    ExtraCursor.transform.LookAt(ChaseCursorObject.transform.position);

    if (Physics.Raycast(viewpointStart, ExtraCursor.transform.forward, out rayHit, 1000.0f, layer2))
    {
        TDMarker.transform.position = rayHit.point;
        Debug.DrawLine(viewpointStart, rayHit.point, Color.red);
    }
    Debug.DrawLine(viewpointStart, viewpoint, Color.white);

    /* Click button via blinking. Do not use if clicking usning hands */
    GameMaster gm = GameMaster.instance;
    if (!gm.isHandTriggered)
    {
        if ((MLEyes.LeftEye.IsBlinking && !MLEyes.RightEye.IsBlinking)
            || (!MLEyes.LeftEye.IsBlinking && MLEyes.RightEye.IsBlinking))
        {
            //if blinking was not on and became on in this frame start clicking
            if (!isBlinking)
            {
                ChaseCursorObject.GetComponent<Image>().color = CursorBlinkingColor;
                if (gm.SelectedButton != null)
                    gm.SelectedButton.DelayedClick();
            }
            isBlinking = true;
        }
        else
        {
            //if blinking was on and became off in this frame stop clicking
            if (isBlinking)
            {
                ChaseCursorObject.GetComponent<Image>().color = CursorIdleColor;
                if (gm.SelectedButton != null)
                    gm.SelectedButton.DelayedClickStop();
            }
            isBlinking = false;
        }
    }
}
```

Figure 98: The Eye Tracking Manager script

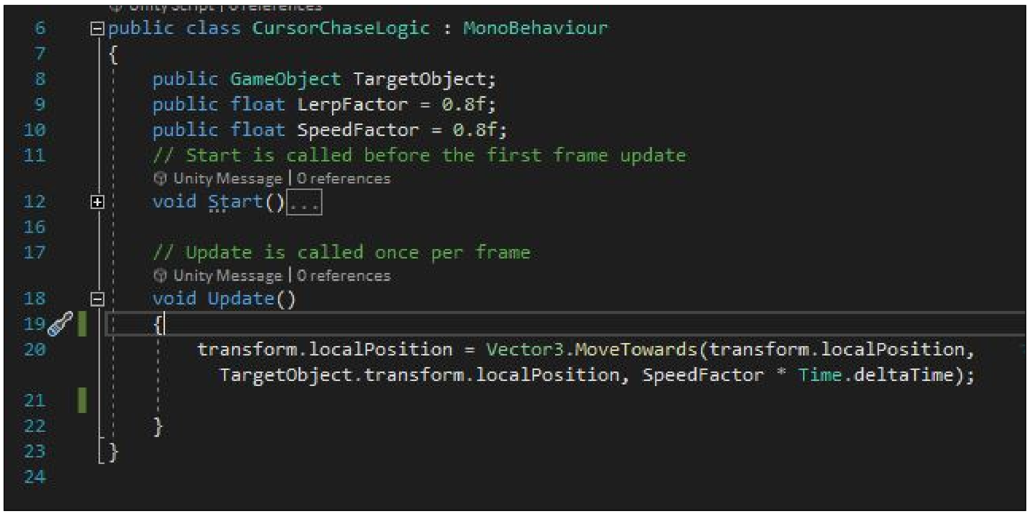
Eye tracking interactions are achieved through the *Eye Tracking Manager* script (see Fig. 98) which is placed on the *Cursor* Object. The eye tracker's fixation point is defined through the

MLEyes class included in the *UnityEngine.XR.MagicLeap* package using the *MLEyes.FixationPoint* parameter (line 40).

The script executes a process on every frame inside the *Update* function. First, we position an *ExtraCursor* object on the *MainCamera*'s position (line 34) and rotate the *ExtraCursor* to look towards the eye tracker's fixation point (line 42). Then we perform a raycast from the *ExtraCursor*'s forward vector which collides only with objects in the "UI" layer (line 36,38) which only includes the *Fixed Canvas* object and then reposition the *Cursor* object on the position the ray hit the *Fixed Canvas.*

Next, we perform a second raycast and this time the *ExtraCursor* is looking towards the *ChaseCursor* which collides with objects inside the "World UI" layer which includes the Glanceable, Tutorial, Holo-box and Evaluation Interfaces and positions the *World cursor* on the hit position (lines 50-60).

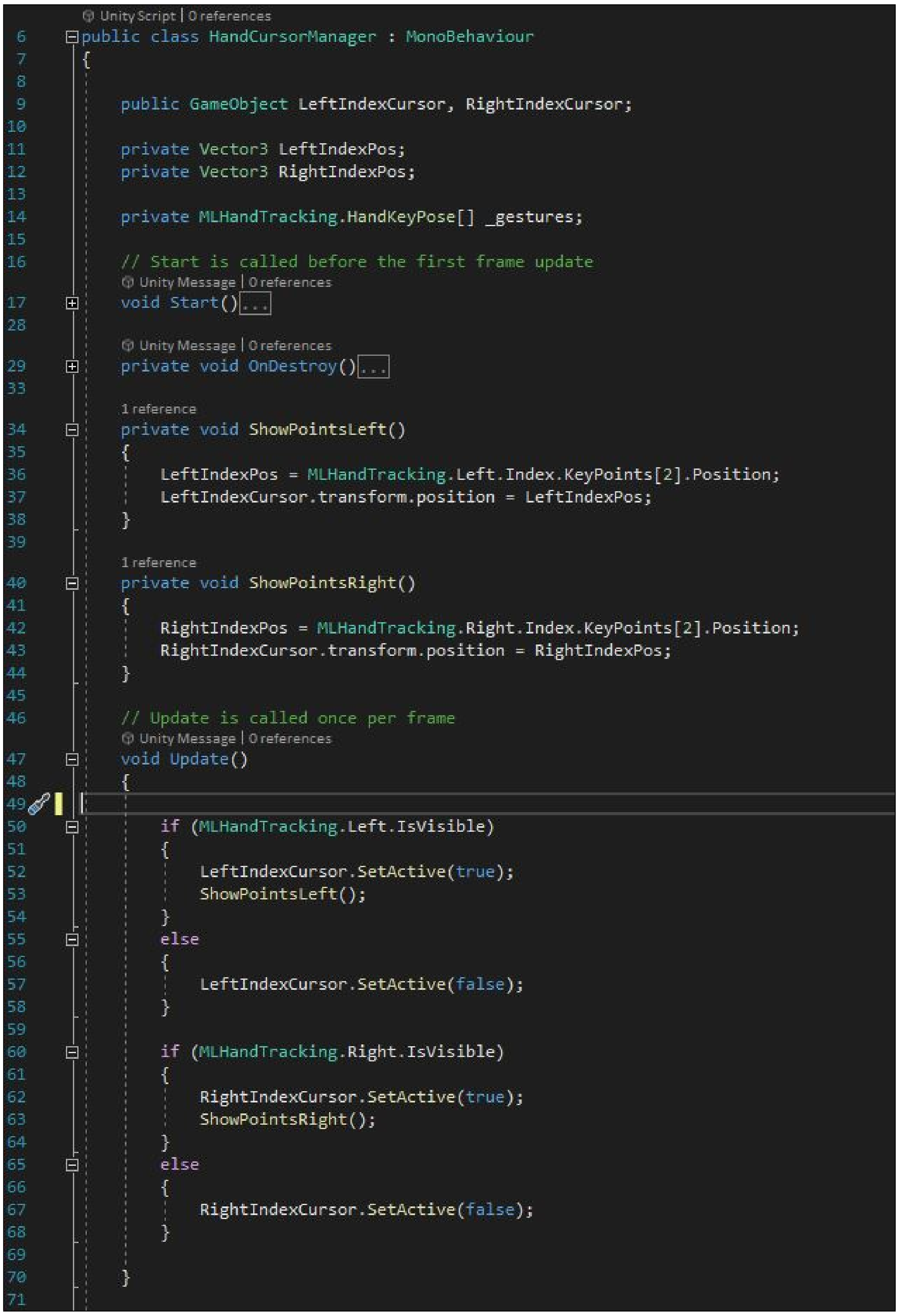
Finally, at the end of the frame, we check the *MLEyes.Left.Eye.IsBlinking* and the *MLEyes.RightEye.IsBlinking* parameters and if the user is blinking with one of their eyes and the user has not selected an object with their hands then a *DelayedClick* is executed. Similarly, if the user was performing a *DelayedClick* using their eyes and they stop blinking the click is topped with the *DelayedClickStop* function (lines 62-89).



```
public class CursorChaseLogic : MonoBehaviour
{
    public GameObject TargetObject;
    public float LerpFactor = 0.8f;
    public float SpeedFactor = 0.8f;
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()...

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        transform.localPosition = Vector3.MoveTowards(transform.localPosition,
          TargetObject.transform.localPosition, SpeedFactor * Time.deltaTime);

    }
}
```

Figure 99: The Chase Cursor Manager script

The *ChaseCursor* handles its own movement through a simple generic script (see Fig. 99) which sets a speed value and a target and then moves towards the target at every frame.



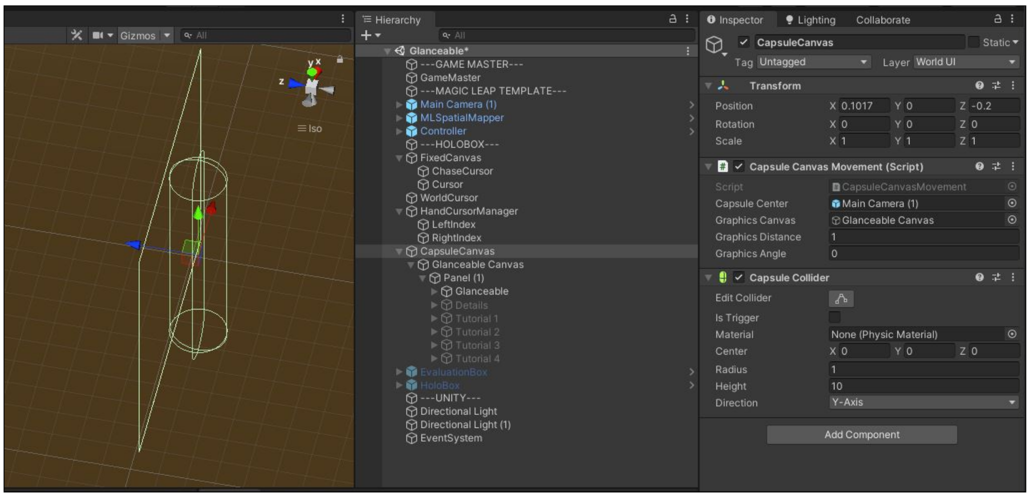
```
Unity Script | 0 references
public class HandCursorManager : MonoBehaviour
{

    public GameObject LeftIndexCursor, RightIndexCursor;

    private Vector3 LeftIndexPos;
    private Vector3 RightIndexPos;

    private MLHandTracking.HandKeyPose[] _gestures;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()...

    Unity Message | 0 references
    private void OnDestroy()...

    1 reference
    private void ShowPointsLeft()
    {
        LeftIndexPos = MLHandTracking.Left.Index.KeyPoints[2].Position;
        LeftIndexCursor.transform.position = LeftIndexPos;
    }

    1 reference
    private void ShowPointsRight()
    {
        RightIndexPos = MLHandTracking.Right.Index.KeyPoints[2].Position;
        RightIndexCursor.transform.position = RightIndexPos;
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

        if (MLHandTracking.Left.IsVisible)
        {
            LeftIndexCursor.SetActive(true);
            ShowPointsLeft();
        }
        else
        {
            LeftIndexCursor.SetActive(false);
        }

        if (MLHandTracking.Right.IsVisible)
        {
            RightIndexCursor.SetActive(true);
            ShowPointsRight();
        }
        else
        {
            RightIndexCursor.SetActive(false);
        }

    }

```

Figure 100: The Hand Cursor Manager script

Hand Cursor Manager (see Fig. 100). The Hand cursors consist of two Game Objects visualised as two small purple spheres (see Fig. 81) which follow the user's index fingers. The spherers have a radius of 0.5 millimeters, which is approximately the same size as the index finger's nail.

The *Hand Cursor Manager* script uses the *MLHandTracking* class from the *UnityEngine.XR.MagicLeap* package to receive tracking data from the Magic Leap's hand tracker. On every frame through the *Update* function the script checks if either hand is visible through the *MLHandTracking.Left.IsVisible* and *MLHandTracking.Right.IsVisible* parameters and if either hand is visible its cursor is enabled and positioned on the position where the index finger is tracked through the *MLHand-*

Tracking.Left.KeyPoints[2].Position and *MLHandTracking.Right.KeyPoints[2].Position* parameters. Magic leap Tracks 15 key points for every hand and *KeyPoints[2]* indicates the key point at the edge of the index finger. As an alternative, we also used *KeyPoints[0]* which is on the center of the palm which was easier to track but interactions with this point felt unnatural.

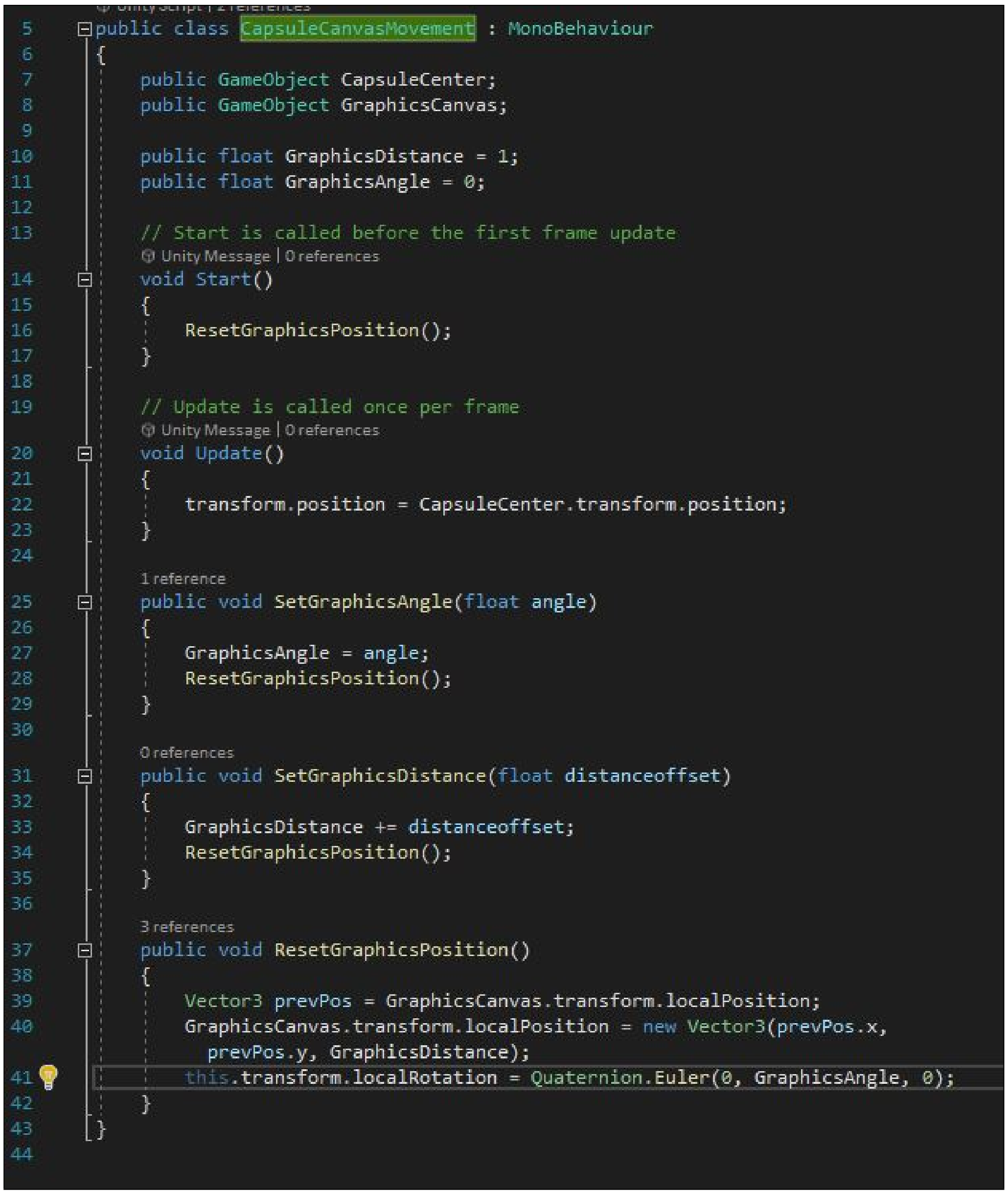


```
Glanceable*
  ---GAME MASTER---
  GameMaster
  ---MAGIC LEAP TEMPLATE---
  Main Camera (1)
  MLSpatialMapper
  Controller
  ---HOLOBOX---
  FixedCanvas
    ChaseCursor
    Cursor
  WorldCursor
  HandCursorManager
    LeftIndex
    RightIndex
  CapsuleCanvas
    Glanceable Canvas
      Panel (1)
        Glanceable
        Details
        Tutorial 1
        Tutorial 2
        Tutorial 3
        Tutorial 4
  EvaluationBox
  HoloBox
  ---UNITY---
  Directional Light
  Directional Light (1)
  EventSystem
```

```
CapsuleCanvas                          Static
Tag  Untagged          Layer  World UI

Transform
Position          X 0.1017   Y 0   Z -0.2
Rotation          X 0        Y 0   Z 0
Scale             X 1        Y 1   Z 1

Capsule Canvas Movement (Script)
Script            CapsuleCanvasMovement
Capsule Center    Main Camera (1)
Graphics Canvas   Glanceable Canvas
Graphics Distance 1
Graphics Angle    0

Capsule Collider
Edit Collider
Is Trigger
Material          None (Physic Material)
Center            X 0   Y 0   Z 0
Radius            1
Height            10
Direction         Y-Axis
```

Figure 101: The Capsule Canvas in the Scene (left), Hierarchy (middle) and Inspector (right)

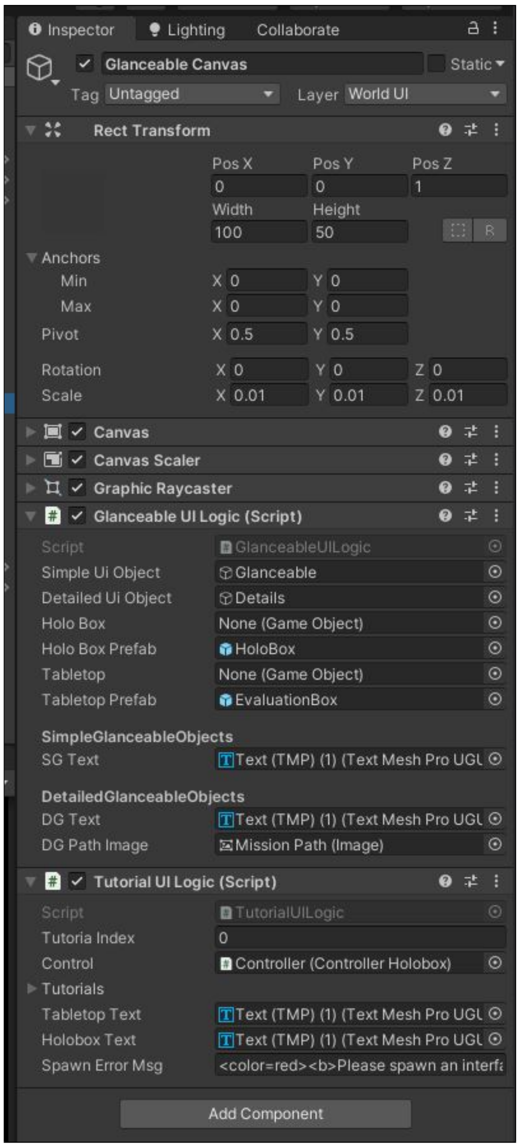
Capsule Canvas (see Fig. 101). The capsule canvas is a combination of two Game objects including the parent *CapsuleCanvas* which is a cylinder centered around the *Main Camera* and a 2D *Glanceable canvas* which rotates around the capsule's center along it's surface.



```
public class CapsuleCanvasMovement : MonoBehaviour
{
    public GameObject CapsuleCenter;
    public GameObject GraphicsCanvas;

    public float GraphicsDistance = 1;
    public float GraphicsAngle = 0;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        ResetGraphicsPosition();
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        transform.position = CapsuleCenter.transform.position;
    }

    1 reference
    public void SetGraphicsAngle(float angle)
    {
        GraphicsAngle = angle;
        ResetGraphicsPosition();
    }

    0 references
    public void SetGraphicsDistance(float distanceoffset)
    {
        GraphicsDistance += distanceoffset;
        ResetGraphicsPosition();
    }

    3 references
    public void ResetGraphicsPosition()
    {
        Vector3 prevPos = GraphicsCanvas.transform.localPosition;
        GraphicsCanvas.transform.localPosition = new Vector3(prevPos.x,
          prevPos.y, GraphicsDistance);
        this.transform.localRotation = Quaternion.Euler(0, GraphicsAngle, 0);
    }
}
```

Figure 102: The Capsule Canvas Movement script

The *Capsule Canvas Movement* script (see Fig. 102) contains functions that set the orientation and position of Glanceable interfaces. On every frame through the *Update* function, the capsule is set to follow the position of its *CapsuleCenter* parameter, which is set to be the position of the *Main Camera* (see Fig. 101). The *SetGraphicsAngle* function is called from the *Controller Holobox* script to adjust the rotation of the capsule canvas during runtime. Similarly, the *SetGraphicsDistance* function can be used to adjust the radius of the capsule canvas, moving the *Glanceable canvas* closer or further from the user.



Glanceable Canvas
Static
Tag Untagged
Layer World UI

Rect Transform

	Pos X	Pos Y	Pos Z
	0	0	1
	Width	Height	
	100	50	
Anchors			
Min	X 0	Y 0	
Max	X 0	Y 0	
Pivot	X 0.5	Y 0.5	
Rotation	X 0	Y 0	Z 0
Scale	X 0.01	Y 0.01	Z 0.01

Canvas
Canvas Scaler
Graphic Raycaster
Glanceable UI Logic (Script)

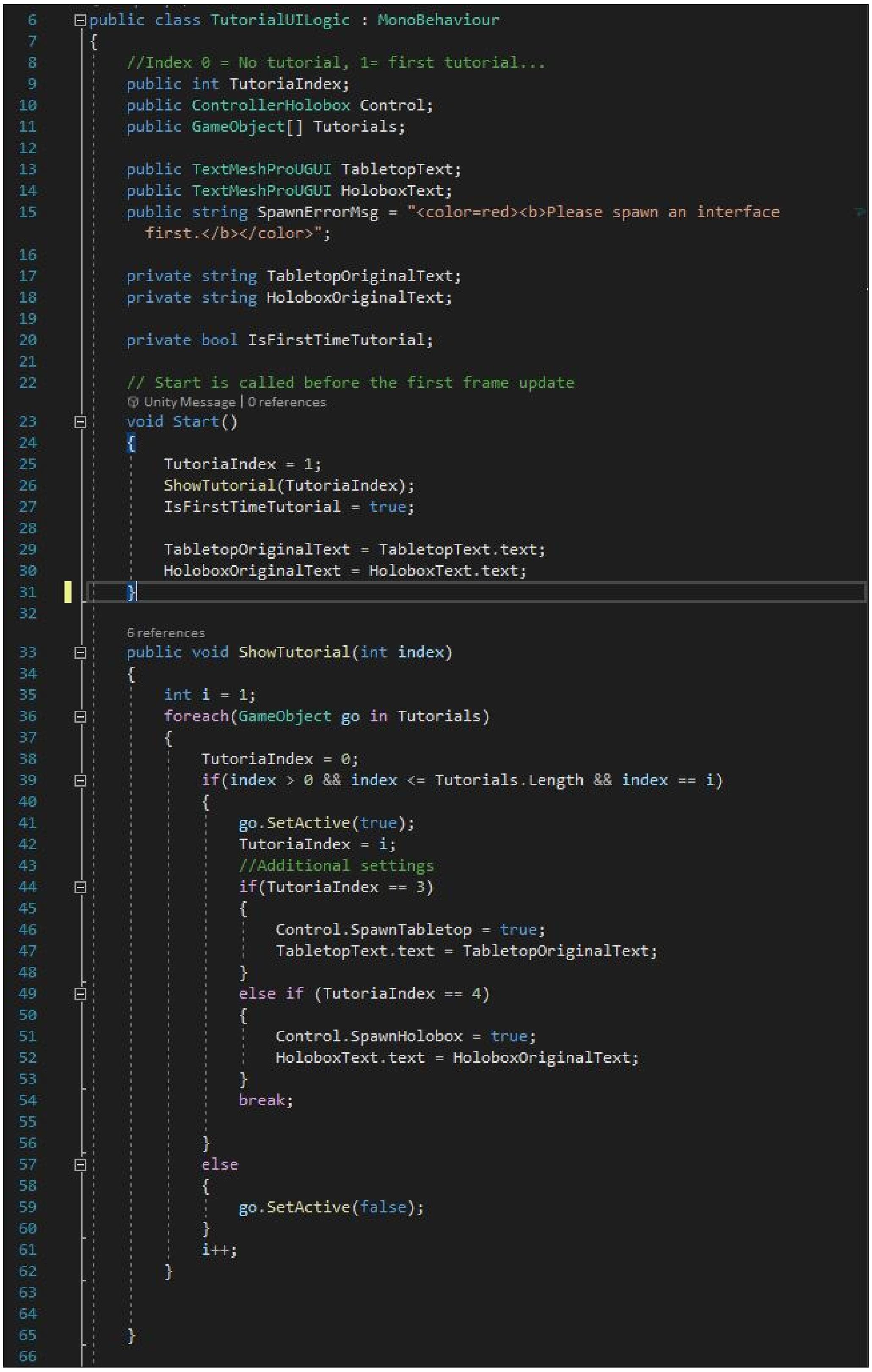
Script	GlanceableUILogic
Simple Ui Object	Glanceable
Detailed Ui Object	Details
Holo Box	None (Game Object)
Holo Box Prefab	HoloBox
Tabletop	None (Game Object)
Tabletop Prefab	EvaluationBox
SimpleGlanceableObjects	
SG Text	Text (TMP) (1) (Text Mesh Pro UGL
DetailedGlanceableObjects	
DG Text	Text (TMP) (1) (Text Mesh Pro UGL
DG Path Image	Mission Path (Image)

Tutorial UI Logic (Script)

Script	TutorialUILogic
Tutoria Index	0
Control	Controller (Controller Holobox)
Tutorials	
Tabletop Text	Text (TMP) (1) (Text Mesh Pro UGL
Holobox Text	Text (TMP) (1) (Text Mesh Pro UGL
Spawn Error Msg	<color=red><b>Please spawn an interfa

Figure 103: The Glanceable Canvas Object's inspector

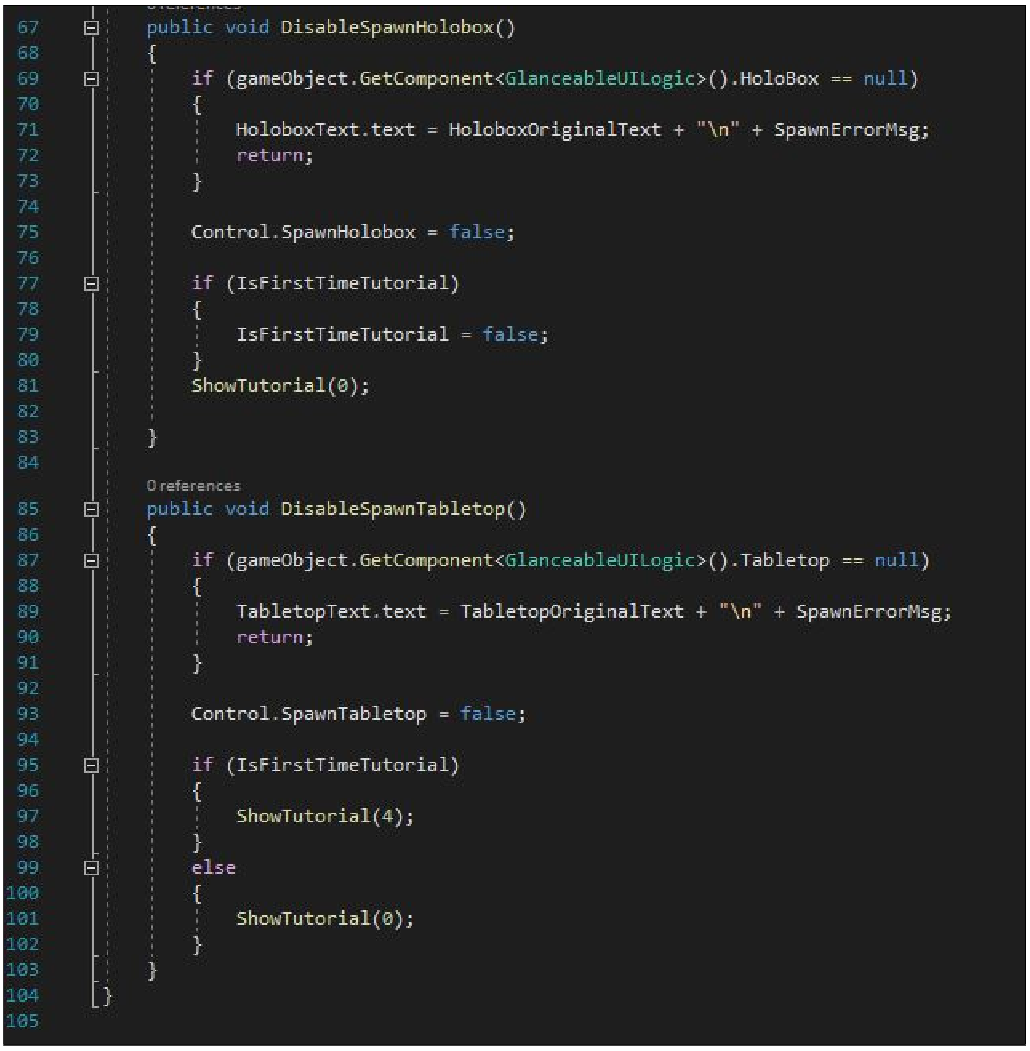
The *Glanceable Canvas* contains all 2D Glanceable interfaces including the *Glanceable* (Simple Glanceable interface), *Details* (Detailed Glanceable interface) and four *Tutorial interfaces* (see Fig. 101) In addition, the *Glanceable Canvas* has two scripts, the *Glanceable UI Logic* for managing the Glanceable interfaces and the *Tutorial UI Logic* script for managing the *Tutorial* interfaces.



```
public class TutorialUILogic : MonoBehaviour
{
    //Index 0 = No tutorial, 1= first tutorial...
    public int TutoriaIndex;
    public ControllerHolobox Control;
    public GameObject[] Tutorials;

    public TextMeshProUGUI TabletopText;
    public TextMeshProUGUI HoloboxText;
    public string SpawnErrorMsg = "<color=red><b>Please spawn an interface
      first.</b></color>";

    private string TabletopOriginalText;
    private string HoloboxOriginalText;

    private bool IsFirstTimeTutorial;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        TutoriaIndex = 1;
        ShowTutorial(TutoriaIndex);
        IsFirstTimeTutorial = true;

        TabletopOriginalText = TabletopText.text;
        HoloboxOriginalText = HoloboxText.text;
    }

    6 references
    public void ShowTutorial(int index)
    {
        int i = 1;
        foreach(GameObject go in Tutorials)
        {
            TutoriaIndex = 0;
            if(index > 0 && index <= Tutorials.Length && index == i)
            {
                go.SetActive(true);
                TutoriaIndex = i;
                //Additional settings
                if(TutoriaIndex == 3)
                {
                    Control.SpawnTabletop = true;
                    TabletopText.text = TabletopOriginalText;
                }
                else if (TutoriaIndex == 4)
                {
                    Control.SpawnHolobox = true;
                    HoloboxText.text = HoloboxOriginalText;
                }
                break;

            }
            else
            {
                go.SetActive(false);
            }
            i++;
        }


    }

```

Figure 104: The Tutorial UI Logic script. Initial tutorial execution

The *Tutorial UI Logic* is executed when the application starts through the *Start* function (see Fig. 104). On start, the logic shows the first tutorial interface which shows the first *Tutorial interface*. Each *Tutorial interface* includes a button with a script that is responsible for increasing the *TutorialIndex* and calling *ShowTutorial* again.

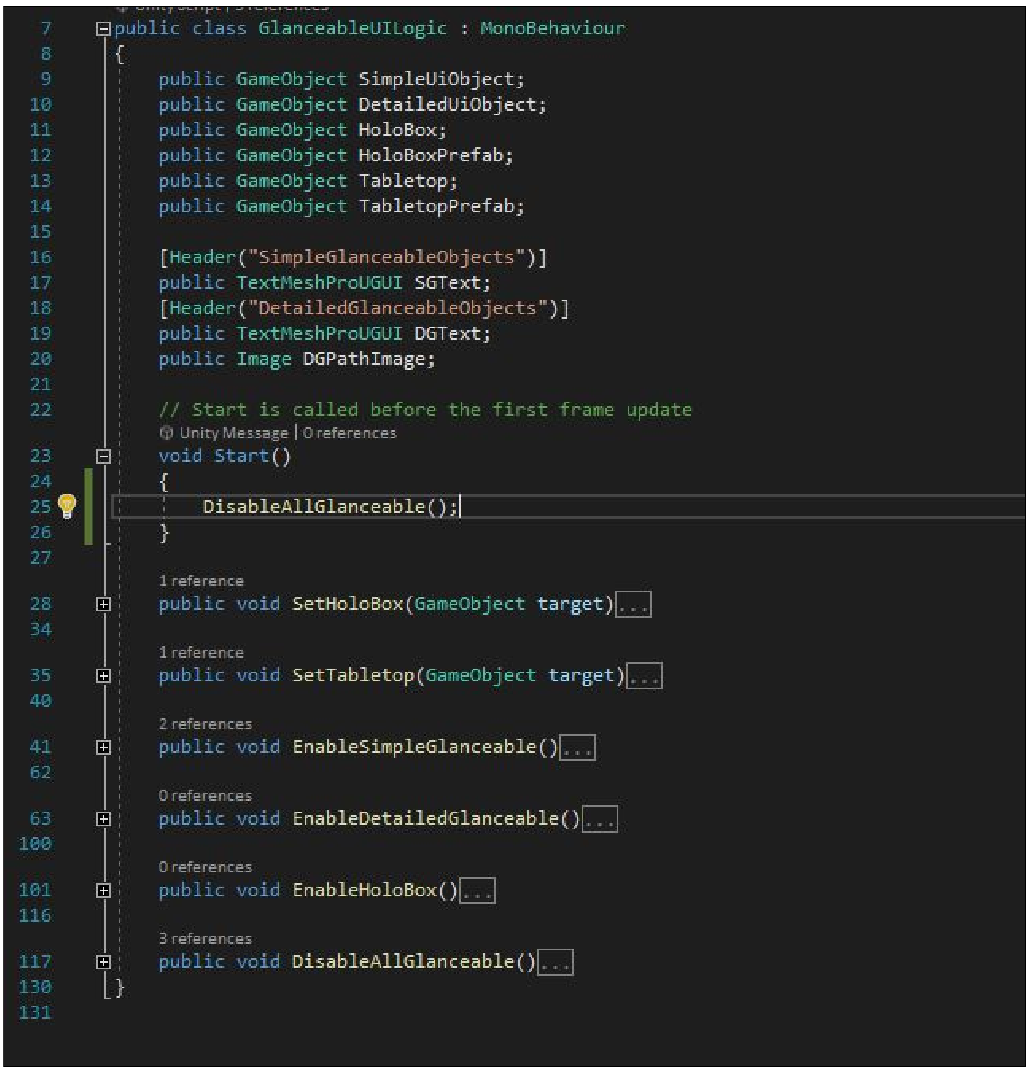


```
public void DisableSpawnHolobox()
{
    if (gameObject.GetComponent<GlanceableUILogic>().HoloBox == null)
    {
        HoloboxText.text = HoloboxOriginalText + "\n" + SpawnErrorMsg;
        return;
    }

    Control.SpawnHolobox = false;

    if (IsFirstTimeTutorial)
    {
        IsFirstTimeTutorial = false;
    }
    ShowTutorial(0);

}

0 references
public void DisableSpawnTabletop()
{
    if (gameObject.GetComponent<GlanceableUILogic>().Tabletop == null)
    {
        TabletopText.text = TabletopOriginalText + "\n" + SpawnErrorMsg;
        return;
    }

    Control.SpawnTabletop = false;

    if (IsFirstTimeTutorial)
    {
        ShowTutorial(4);
    }
    else
    {
        ShowTutorial(0);
    }
}
}
```

Figure 105: The Tutorial UI Logic script. Reposition interfaces

The third and fourth *Tutorial Interfaces* are responsible for spawning the *Evaluation Interface* and *Holo-Box interface* objects. Thus, when their buttons are pressed, they call the *DisableSpawnTabletop* and *DisableSpawnHolobox* functions accordingly (see Fig. 105). These functions check if the appropriate interface was spawned and then proceed to the next tutorial step. The *TutorialIndex* is also used as an indicator on the *Controller Holobox* script to determine which interface to spawn when the Magic Leap Control bumper is clicker. Additionally, the Glanceable, evaluation and Holo-box interfaces are disabled as long as the *TutorialIndex* is not zero.

These interfaces may also be activated again through the *Evaluation Interface* after the tutorial is completed in order to reposition the *Evaluaiton Interface* or *Holo-Box interface*. In this case, the tutorials will not be executed sequentially and instead after one tutorial step the *TutorialIndex* will be reset to zero indicating no *Tutorial interface* is enabled.



```
public class GlanceableUILogic : MonoBehaviour
{
    public GameObject SimpleUiObject;
    public GameObject DetailedUiObject;
    public GameObject HoloBox;
    public GameObject HoloBoxPrefab;
    public GameObject Tabletop;
    public GameObject TabletopPrefab;

    [Header("SimpleGlanceableObjects")]
    public TextMeshProUGUI SGText;
    [Header("DetailedGlanceableObjects")]
    public TextMeshProUGUI DGText;
    public Image DGPathImage;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        DisableAllGlanceable();
    }

    1 reference
    public void SetHoloBox(GameObject target)...

    1 reference
    public void SetTabletop(GameObject target)...

    2 references
    public void EnableSimpleGlanceable()...

    0 references
    public void EnableDetailedGlanceable()...

    0 references
    public void EnableHoloBox()...

    3 references
    public void DisableAllGlanceable()...
}
```

Figure 106: The Glanceable UI Logic script

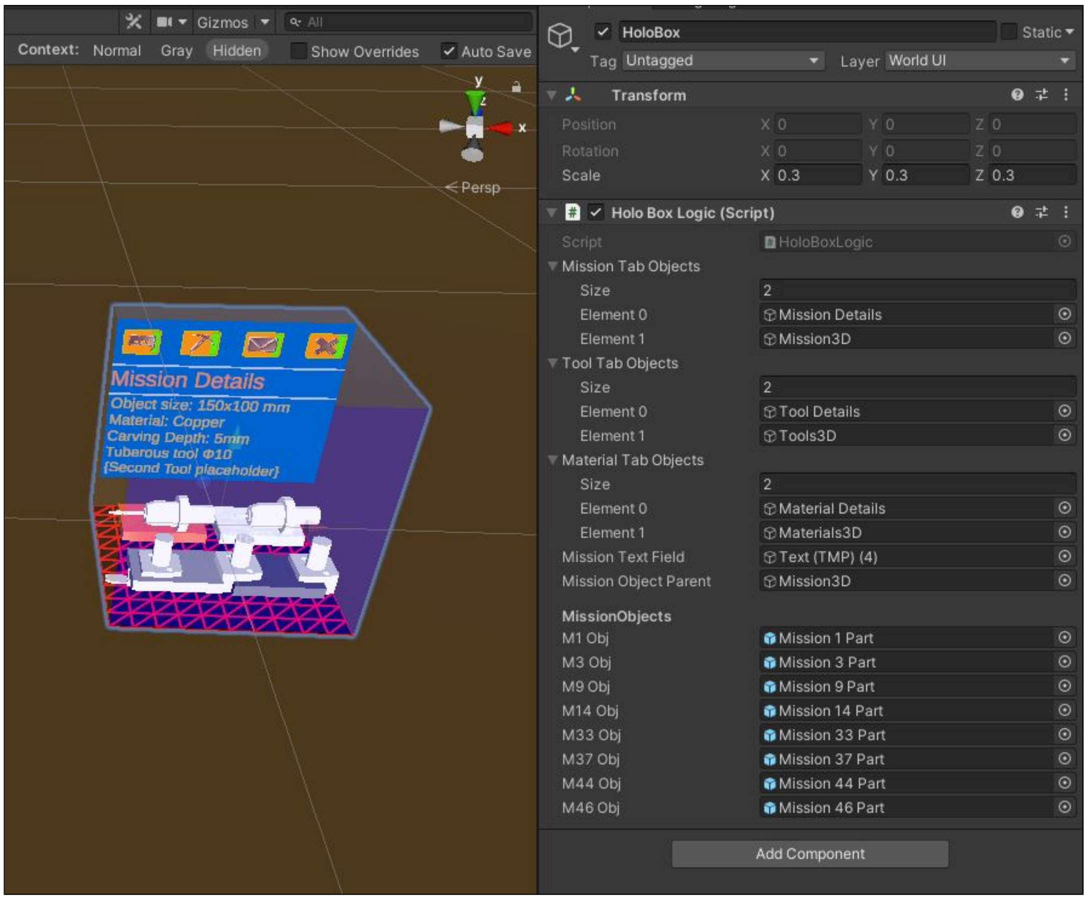
The *Glanceable UI Logic* script handles all logic regarding enabling and disabling the Glanceable and Holo-box interfaces. At the start of the application's execution through the *Start* function, all three interfaces are disabled.

When the Magic Leap's Control's bumper is pressed through the *Controller Holobox* script the appropriate interface is instantiated and then the *SetHoloBox* or *SetTabletop* functions are called, destroying the previous instance of the object, if any, and then setting the *HoloBox* or *Tabletop* parameters to the new object.

The *EnableSimpleGlanceable*, *EnableDetailedGlanceable* and *EnableHoloBox* functions enable the appropriate interface while disabling the other two. These functions also set the objects and values that appear in each interface according to the selected mission's parameters including setting texts, the 2D image of the Detailed Glanceable interface and the 3D model of the Holo-box's finished product.

These three functions are called at the following times:

- *EnableSimpleGlanceable*: When a new mission is activated through the *Evaluation Box Logic*. When the Holo-box interface is closed using the "X" button.
- *EnableDetailedGlanceable* When the "eye" button on the Simple Glanceable interface is pressed.
- *EnableHoloBox* When the "eye" button on the Detailed Glanceable interface is pressed.



Mission Details
Object size: 150x100 mm
Material: Copper
Carving Depth: 5mm
Tuberous tool Φ10
{Second Tool placeholder}

HoloBox
Tag Untagged Layer World UI

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 0.3	Y 0.3	Z 0.3

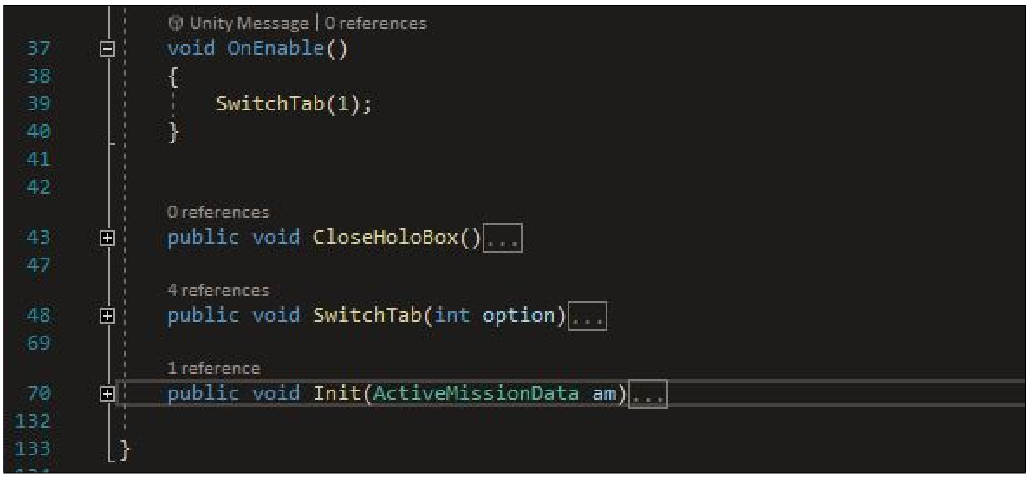
Holo Box Logic (Script)

Script	HoloBoxLogic
Mission Tab Objects	
Size	2
Element 0	Mission Details
Element 1	Mission3D
Tool Tab Objects	
Size	2
Element 0	Tool Details
Element 1	Tools3D
Material Tab Objects	
Size	2
Element 0	Material Details
Element 1	Materials3D
Mission Text Field	Text (TMP) (4)
Mission Object Parent	Mission3D
MissionObjects	
M1 Obj	Mission 1 Part
M3 Obj	Mission 3 Part
M9 Obj	Mission 9 Part
M14 Obj	Mission 14 Part
M33 Obj	Mission 33 Part
M37 Obj	Mission 37 Part
M44 Obj	Mission 44 Part
M46 Obj	Mission 46 Part

Figure 107: The Holo-Box Interface in the Scene (left) and Inspector (right)

Holo-Box Interface (see Fig. 107). The Holo-box holds multiple 2D and 3D objects which includes the following:

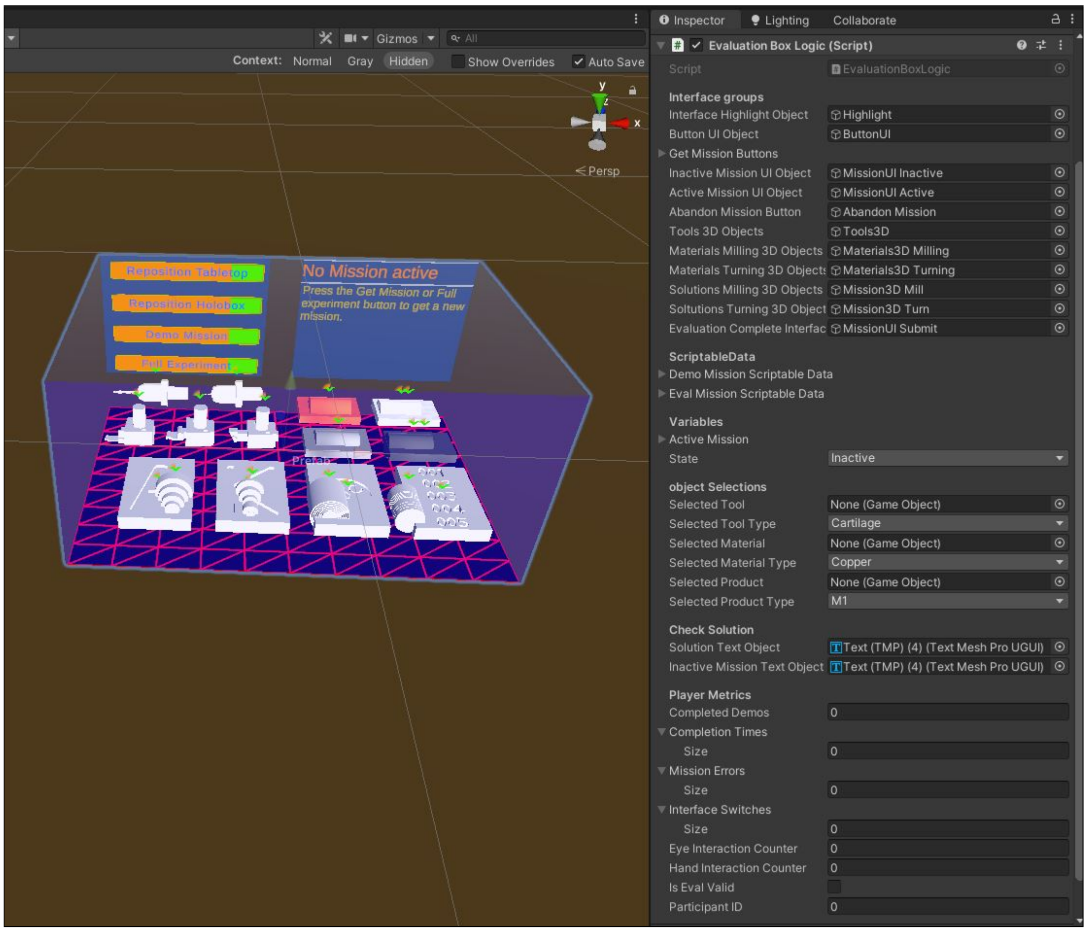
- Floor panel. This is indicated by a wireframe pattern at the bottom of the Holo-box
- Transparent box. This is the blue box which indicates the perimeter inside which virtual content will appear. The box is always visible at 95% transparency making it visible without occluding its contents.
- Back Panel. The back panel consists of two parts. On the top of the panel are four interactable buttons. The first three cycle between the three "tabs" of the Holo-Box interface including the *Mission Tab* (see Fig. 83), *Tool Tab* (see Fig. 84a) and *Materials Tab* (see Fig. 84a) with the fourth button closing the Holo-Box interface and returning to the Simple Glanceable. Below the buttons is a description whose content changes based on which tab is enabled.
- 3D tools. Five tools are placed inside the Holo-box, which are shown when the *Tool Tab* is enabled. Each tool includes a 3D text with its name which is placed above the 3D model and is shown when the tool is selected using eye or hand cursors.
- 3D Materials. Four Rectangular blocks indicated four different materials are placed inside the Holo-box, which are shown when the *Material Tab* is enabled. Each material includes a 3D text with its name which is placed above the 3D model and is shown when the material is selected using eye or hand cursors.
- 3D Finished Mission Product. Unlike the Tools and materials, which were manually placed so that their colliders don't overlap, Holo-box includes an array of references to all finished products inside the *MissionObjects* list. When the *Mission Tab* is shown, the appropriate finished product is instantiated during runtime.



```
    Unity Message | 0 references
    void OnEnable()
    {
        SwitchTab(1);
    }


    0 references
    public void CloseHoloBox()...

    4 references
    public void SwitchTab(int option)...

    1 reference
    public void Init(ActiveMissionData am)...
}
```

Figure 108: The Holobox UI Logic script

Initially, the Holo-box is enabled when it is spawned during the Tutorial. During that time, all the objects except the *Floor panel* and *Transparent box* are disabled. When the "eye" button of the Detailed Glanceable interface is pressed the content of the Holo-box is enabled, showing the *Mission Tab* (see Fig. 108) on the *OnEnable* function. The buttons that change the active tab execute the *SwitchTab* function with the appropriate parameter. If the Holo-box's "X" button is pressed, all tabs are disabled and the *CloseHoloBox* function is called, enabling the Simple Glanceable interface. When a new mission is given to the user, the *Init* function is called setting the content of the *Mission Tab* accordingly.



Evaluation Box Logic (Script)

Script	EvaluationBoxLogic
Interface groups	
Interface Highlight Object	Highlight
Button UI Object	ButtonUI
Get Mission Buttons	
Inactive Mission UI Object	MissionUI Inactive
Active Mission UI Object	MissionUI Active
Abandon Mission Button	Abandon Mission
Tools 3D Objects	Tools3D
Materials Milling 3D Objects	Materials3D Milling
Materials Turning 3D Objects	Materials3D Turning
Solutions Milling 3D Objects	Mission3D Mill
Soltutions Turning 3D Object	Mission3D Turn
Evaluation Complete Interfac	MissionUI Submit
ScriptableData	
Demo Mission Scriptable Data	
Eval Mission Scriptable Data	
Variables	
Active Mission	
State	Inactive
object Selections	
Selected Tool	None (Game Object)
Selected Tool Type	Cartilage
Selected Material	None (Game Object)
Selected Material Type	Copper
Selected Product	None (Game Object)
Selected Product Type	M1
Check Solution	
Solution Text Object	Text (TMP) (4) (Text Mesh Pro UGUI)
Inactive Mission Text Object	Text (TMP) (4) (Text Mesh Pro UGUI)
Player Metrics	
Completed Demos	0
Completion Times	
Size	0
Mission Errors	
Size	0
Interface Switches	
Size	0
Eye Interaction Counter	0
Hand Interaction Counter	0
Is Eval Valid	
Participant ID	0

Figure 109: The Evaluation Interface in the Scene (left) and Inspector (right)

Evaluation Interface (see Fig. 109). The evaluation interface holds all the logic used for the execution of the user study outside the core logic of our proposed interface system.

The prefab of the *Evaluation Interface* includes the following Game Objects:

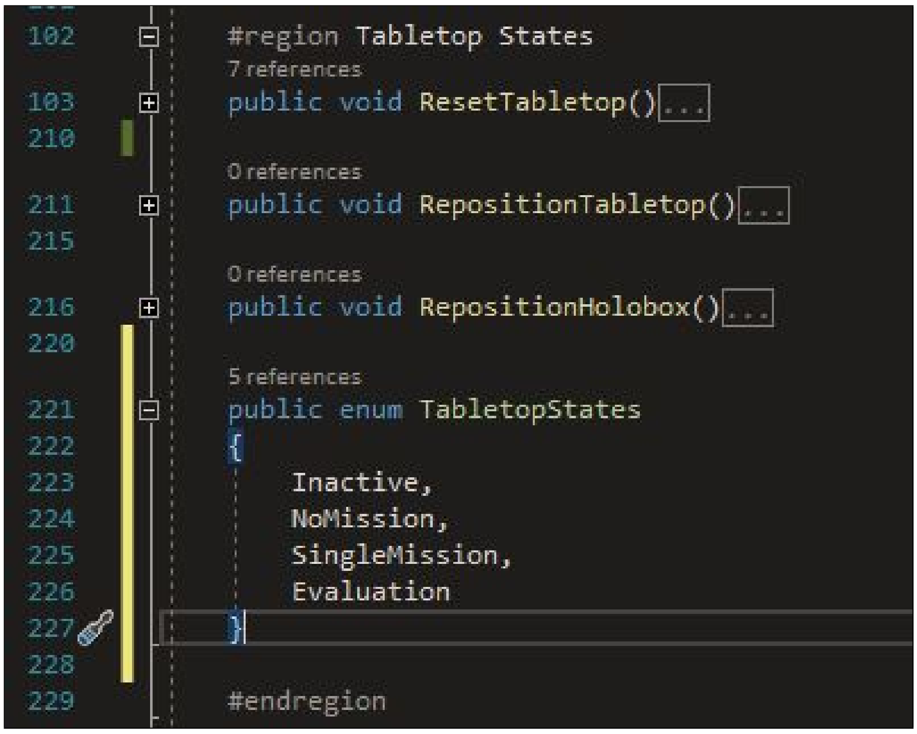
- Floor panel. This is indicated by a wireframe pattern at the bottom of the interface.
- Transparent box. This is the blue box which indicates the perimeter inside which virtual content will appear. The box is always visible at 95% transparency making it visible without occluding its contents.
- Left Back Panel. The Left back panel includes four buttons. The first two buttons include the *Reposition Tabletop* and *Reposition Holobox* buttons, which execute the appropriate logic from the *Tutorial UI Logic* and reposition one of the 3D interfaces. The last two buttons include the *Demo Mission* and *Full Experiment* buttons. The *Demo Mission* button generates a single mission, which can be abandoned at any time and if it is solved or abandoned it returns the Evaluation interface it its default state. The *Full experiment* on the other hand generates 10 missions, which are given to the user back to back and cannot be abandoned and when all 10 missions are completed the Evaluation interface returns to its default state.
- Right Back panel. The second back panel handles the per-mission execution. By default it includes a text stating that "No mission is active". If the user receives a *Demo Mission*, two buttons are added to the interface, one to *Solve Mission* and one to *Abandon mission*. Similarly, if the *Full Experiment* is pressed, only the *Solve mission* button is shown. When the *Solve mission* button is pressed, the Evaluation interface checks whether the use has selected the appropriate 3D objects and completes the mission if successful or provides textual feedback above the buttons if the user made an error.
- 3D tools. Five tools are placed inside the interface, which are shown when a mission is active. Each tool includes an interaction collider and two visual indicators above it, one circularly loading indicator showing that a *Delayed Click* is being executed to select the object and a green arrow to indicate which tool is selected, if any.
- 3D Materials. Four Rectangular or cylindrical objects that indicate four different materials are placed inside the interface, which are shown when a mission is active. The rectangular materials are shown for milling missions that use a rectangular stock material and the cylindrical models are used for turning missions. Each of the four materials is identified by its color and the same four colors are used in both cases. Each material includes an interaction collider and two visual indicators above it, one circularly loading indicator showing that a *Delayed Click* is being executed to select the object and a green arrow to indicate which material is selected, if any.
- 3D Finished product. Four Rectangular or cylindrical objects that indicate eight different Finished products are placed inside the interface, which are shown when a mission is active. The rectangular Finished products are shown for milling missions that use a rectangular stock material and the cylindrical models are used for turning missions. Each Finished product includes an interaction collider and two visual indicators above it, one circularly loading indicator showing that a *Delayed Click* is being executed to select the object and a green arrow to indicate which Finished product is selected, if any.

The Evaluation interface includes the *EvaluationBoxLogic* script that holds multiple variables separated into the following categories (see Fig. 109):

- Interface Groups. This includes references to all the objects ore sets of objects included in the interface.
- Scriptable Data. This includes two arrays of Scriptable Data objects, with each one holding all of the necessary information for each available mission. These are separated into *Demo Missions* and *Evaluation Missions* with three demo and four evaluation missions available.
- Variable Data. These include parameters which are used during runtime. These include the *Active mission* which is currently being executed and the *Tabletop State* enumeration which defines what objects are enabled inside the Evaluation interface at any moment.

- Check solution parameters. These define the text fields which are used to provide feedback after the *Check Solution* button is pressed.
- Player Metrics. These parameters include all the metrics that are written in a .CSV file in sequential format regarding a user's performance during the *Full Experiment*.

The *EvaluationBoxLogic* also includes multiple functions that are used during the execution of the application as follows:

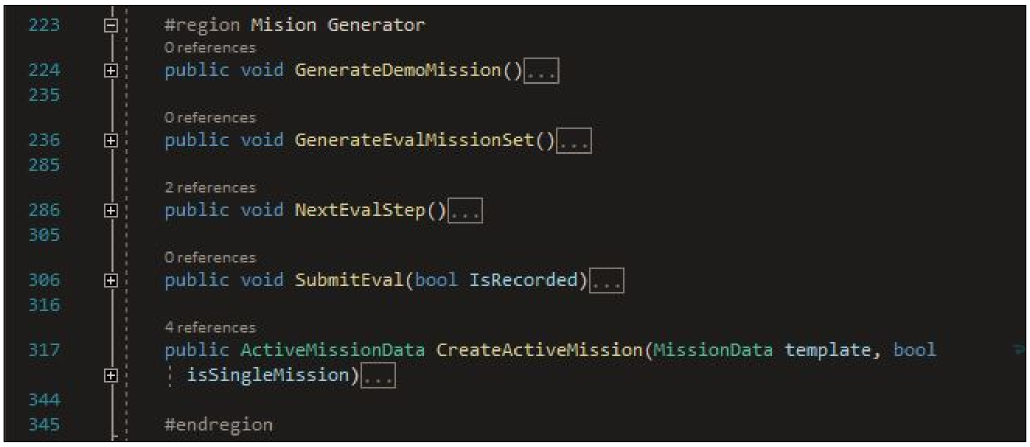


```
#region Tabletop States
7 references
public void ResetTabletop()...

0 references
public void RepositionTabletop()...

0 references
public void RepositionHolobox()...

5 references
public enum TabletopStates
{
    Inactive,
    NoMission,
    SingleMission,
    Evaluation
}

#endregion
```

Figure 110: The Evaluation UI states functions

The *TabletopStates* enumeration defines the four distinct states the Evaluation interface is in at any moment. As long as any tutorial interface is active, the Evaluation Interface is in *Inactive* state. If no tutorial interfaces are active and there is no active mission, the Evaluation Interface is in *NoMission* state. The Evaluation Interface is in the *SingleMission* state if the Active mission is a *Demo mission* or in the *Evaluation* state otherwise.

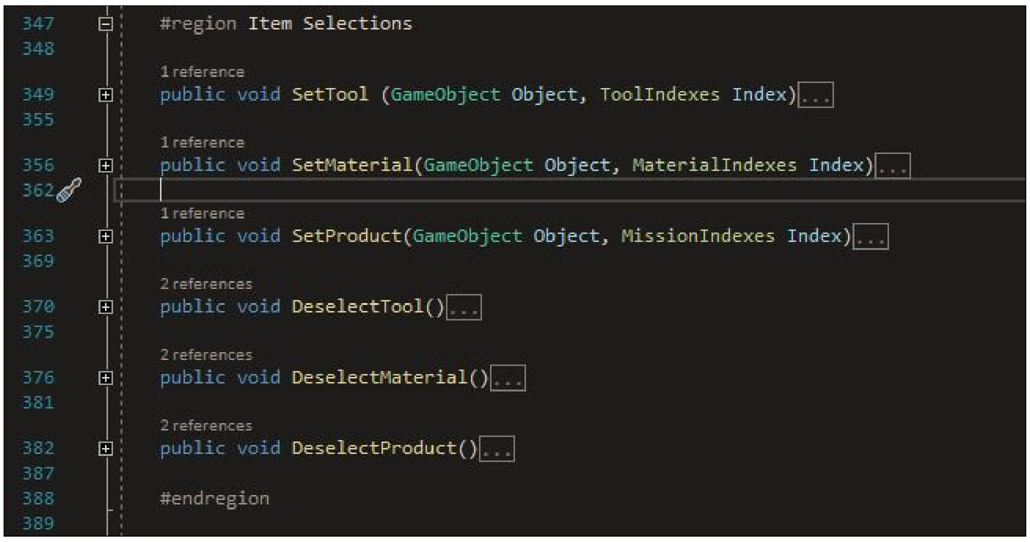
The *RepositionTabletop* and *RepositionHolobox* functions are called when the same buttons are pressed on the Left Back Panel. The *ResetTabletop* is triggered when the *Tabletop State* changes to *NoMission*.



```
#region Mision Generator
0 references
public void GenerateDemoMission()...

0 references
public void GenerateEvalMissionSet()...

2 references
public void NextEvalStep()...

0 references
public void SubmitEval(bool IsRecorded)...

4 references
public ActiveMissionData CreateActiveMission(MissionData template, bool
    isSingleMission)...

#endregion
```

Figure 111: The Evaluation UI mission generator functions

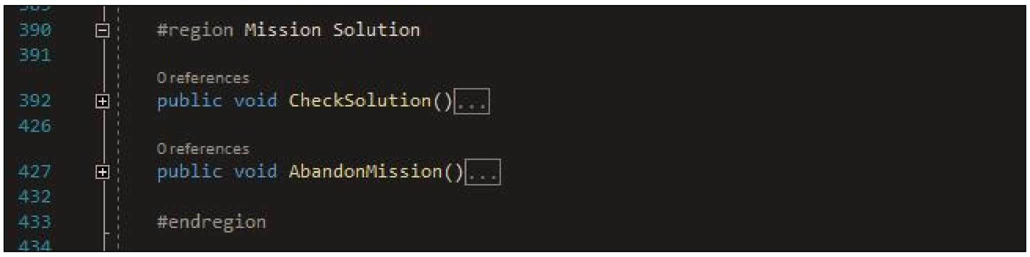
The **GenerateDemoMission** and *GenerateEvalMission* functions are called when the the same buttons are pressed on the Left Back Panel. Both of these functions select one *Mission Data* Scriptable Object from either the *DemoMissionsScriptableData* or *EvalMissionsScriptableData* lists and call the *CreateActiveMission* function with the correct parameters. Similarly, for a *Full experiment* after the first mission is solved, for every subsequent mission the *NextEvalStep* function is called which selects the appropriate *Mission Data* Scriptable Object from the *EvalMissionsScriptableData* list and calls the *CreateActiveMission* function with the correct parameters. The *NextEvalStep* function also writes the *Player Metrics* to a .CSV file and returns the state to *NoMission* if the final mission of the evaluation is completed.



```
#region Item Selections

1 reference
public void SetTool (GameObject Object, ToolIndexes Index)...

1 reference
public void SetMaterial(GameObject Object, MaterialIndexes Index)...

1 reference
public void SetProduct(GameObject Object, MissionIndexes Index)...

2 references
public void DeselectTool()...

2 references
public void DeselectMaterial()...

2 references
public void DeselectProduct()...

#endregion
```

Figure 112: The Evaluation UI item selection functions

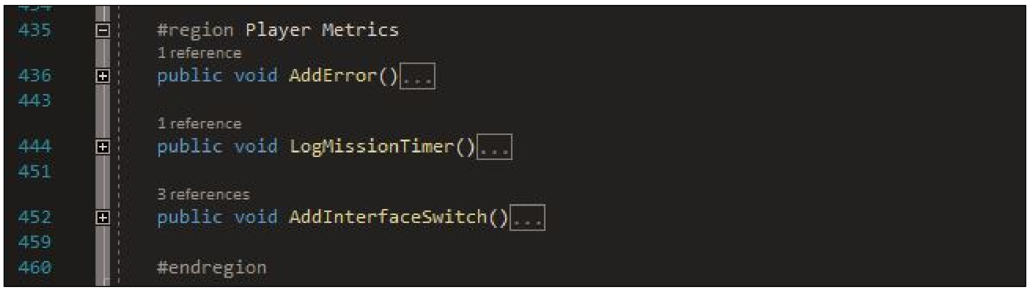
The *SetTool*, *SetMaterial* and *SetProduct* functions toggle the selection of the appropriate tool, material or finished product, deselecting the previous one if any and then enabling it's visual indicator that it is selected. Similarly, the *DeselectTool*, *DeselectMaterial* and *DeselectProduct* functions disable the visual indicator that an object is selected and setting the appropriate value to null. The Deselect functions are called when another object is set or when a mission is solved.



```
#region Mission Solution

0 references
public void CheckSolution()...

0 references
public void AbandonMission()...

#endregion
```

Figure 113: The Evaluation UI mission solution functions

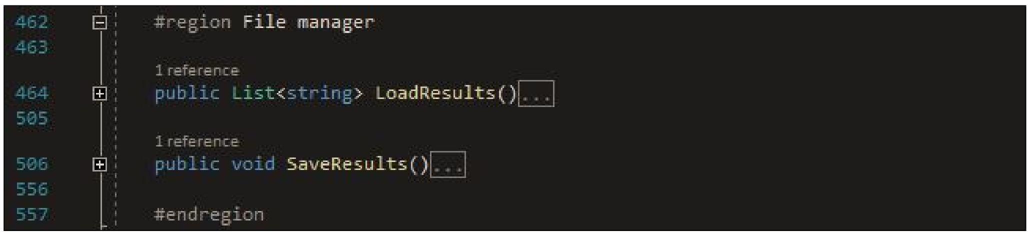
The *CheckSolution* and *AbandonMission* functions are called when the the same buttons are pressed on the Right Back Panel. The *Check solution* function checks which 3D objects are selected in the evaluation interface and then writes feedback on the Right Back Panel if the user made an error, otherwise it records the appropriate data for the mission completion in the *Player metrics* and then returns the state to *NoMission* for a *Demo mission* or calls the *NextEvalStep* for a *Full experiment*. The *Abandon mission* on the other hand sets the *Active Mission* to null and sets the state to *NoMission*.



```
#region Player Metrics
1 reference
public void AddError()...

1 reference
public void LogMissionTimer()...

3 references
public void AddInterfaceSwitch()...

#endregion
```

Figure 114: The Evaluation UI player metrics tracker functions

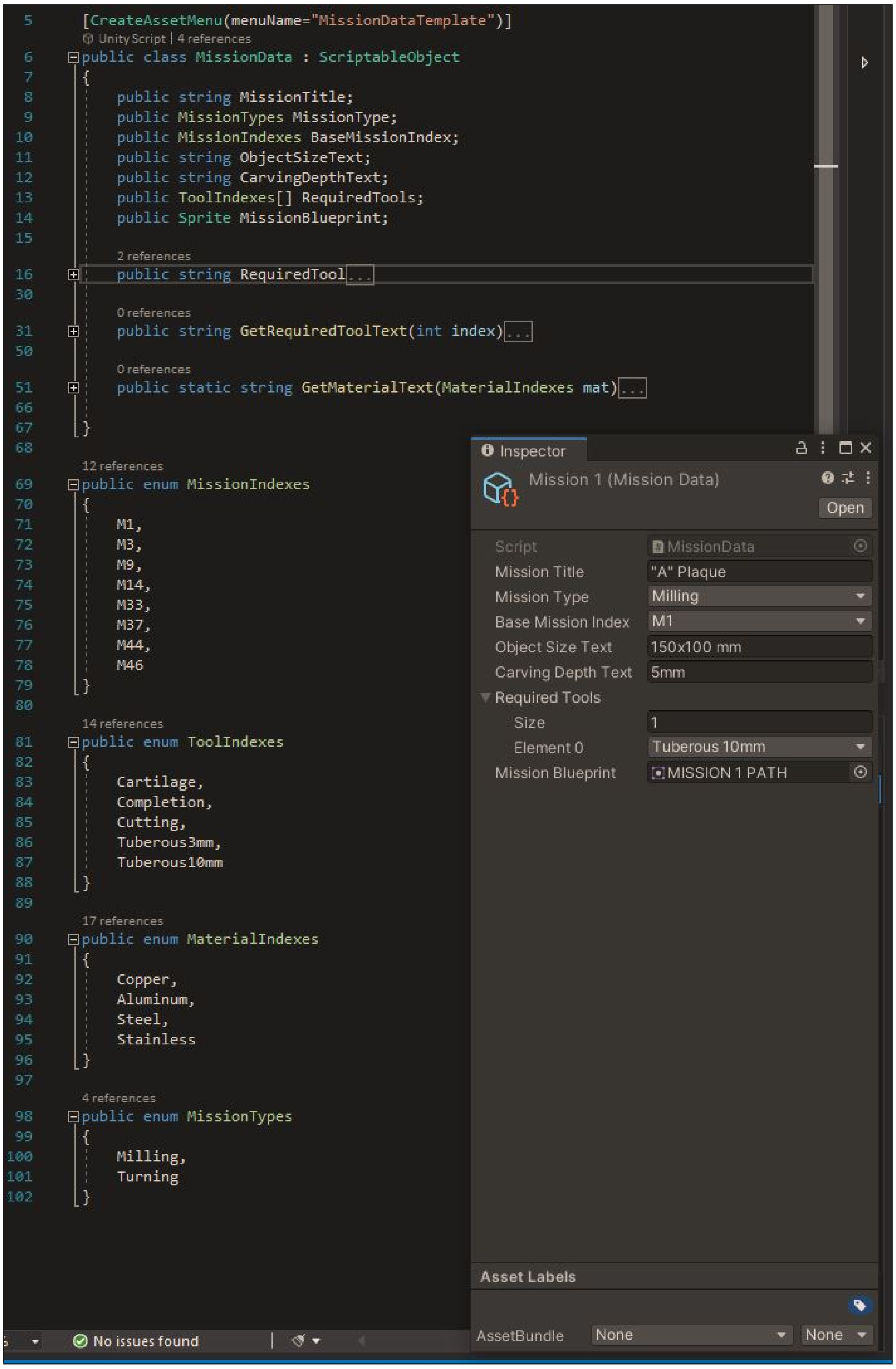
The *AddError*, *LogMissionTimer* and *AddInterfaceSwitch* functions are called from the *CheckSolution* function to record the three parameters which are recorded separately for every mission.



```
#region File manager

1 reference
public List<string> LoadResults()...

1 reference
public void SaveResults()...

#endregion
```

Figure 115: The Evaluation UI file manager functions

The *LoadResults* decrypts the results file from the .CSV format into a list of strings for every line, while the *SaveResults* takes a list of strings and encrypts it as a .CSV file. These functions are called on the *NextEvalStep* function after all the missions of a *Full Experiment* are completed.



```
    [CreateAssetMenu(menuName="MissionDataTemplate")]
    Unity Script | 4 references
public class MissionData : ScriptableObject
{
    public string MissionTitle;
    public MissionTypes MissionType;
    public MissionIndexes BaseMissionIndex;
    public string ObjectSizeText;
    public string CarvingDepthText;
    public ToolIndexes[] RequiredTools;
    public Sprite MissionBlueprint;

    2 references
    public string RequiredTool...

    0 references
    public string GetRequiredToolText(int index)...

    0 references
    public static string GetMaterialText(MaterialIndexes mat)...

}

12 references
public enum MissionIndexes
{
    M1,
    M3,
    M9,
    M14,
    M33,
    M37,
    M44,
    M46
}

14 references
public enum ToolIndexes
{
    Cartilage,
    Completion,
    Cutting,
    Tuberous3mm,
    Tuberous10mm
}

17 references
public enum MaterialIndexes
{
    Copper,
    Aluminum,
    Steel,
    Stainless
}

4 references
public enum MissionTypes
{
    Milling,
    Turning
}
```

No issues found

Inspector

Mission 1 (Mission Data)

Open

Script	MissionData
Mission Title	"A" Plaque
Mission Type	Milling
Base Mission Index	M1
Object Size Text	150x100 mm
Carving Depth Text	5mm
Required Tools	
Size	1
Element 0	Tuberous 10mm
Mission Blueprint	MISSION 1 PATH

Asset Labels

AssetBundle	None	None

Figure 116: The Gamified Manufacturing Mission Scriptable Data Script and Inspector

Gamified Manufacturing Mission Scriptable Data (see Fig. 116). The Scriptable Data class is a special Data holder class type used by Unity. A Scriptable Data class defines a set of variables, an instance of which can be saved on the project's files on the disk as a preset set of values.

Our *MissionData* class includes all of the required information for any given mission. The *MissionType*, *BaseMissionIndex*, *ToolIndex* and the *Material Indexes* are defines as enumerations, allowing for specific values, which can be selected through Unity's inspector as dropdown menus. Even though the *MaterialIndexes* are defined as an enumeration with four values, each mission does not have a predefined material, and the requested material is randomly selected out of the available four every time the *GenerateMission* function is called.

The tool, material, finished product and mission type enumerations are also used as identifiers of their 3D models in the Holo-box and Evaluation Interfaces to identify the correct objects to spawn during their initialization and to check if the correct objects are selected during the *CheckSolution* function.

The *MissionData* class also includes two converters that convert the tool and material enumerations into their text, which is then integrated into the 3-LOD interfaces. The *RequiredTools* are stored as an array if for an extension of this work a mission with multiple required tools is presented.

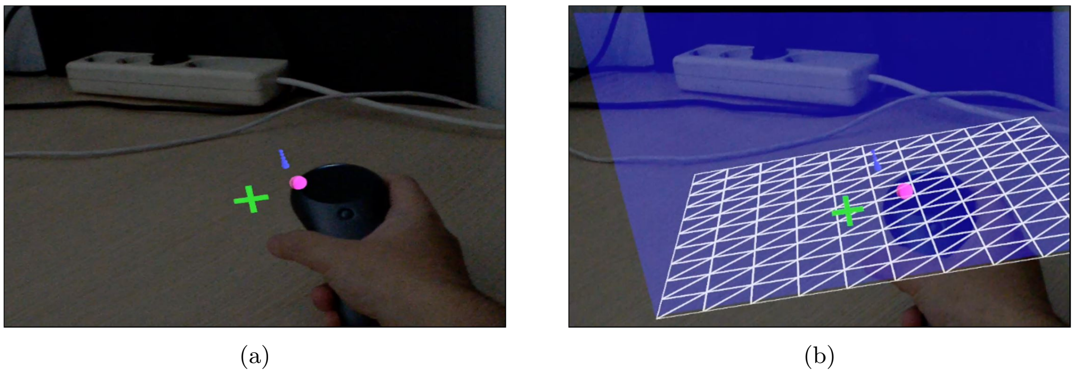
Outside the enumerations, a set of strings are also present including the *MissionTitle*, *ObjectSize* and *CarvingDepth* which are used on the interfaces as-is. Finally, the scriptable data also includes a sprite value which holds the 2D image shown in the Detailed Glanceable UI.

8 User study

A study evaluated the efficiency and effectiveness of our Holo-box interface system in a real-world machine shop. Our first goal is to check the efficiency of our proposed interfaces and interactions under real conditions. Our second goal is to determine the effectiveness of the content of our Glanceable interfaces is adequate when guiding more experienced users without using the most complex HB interface.

8.1 Apparatus

The study used a Magic Leap One AR HMD [74]. The Magic Leap has a 1280x960 resolution and a 50-degree diagonal field of view per eye. The interactions used the built-in hand and eye trackers and the blinking detection provided by the HMD without any perceived latency. The Magic Leap Control controller allowed for the initial setup and the placement of the interfaces, including the rotation of the Glanceable interfaces and the position of the Holo-Box and EI (see Fig. 117).



(a) (b)

Figure 117: Placing interfaces using the Magic Leap Control

The HMD is connected to an external processing and battery unit which was connected to a laptop via cable in order to provide video streaming of the participant's view to a supervisor through the Magic Leap Device Streaming application. Participants performed the experiment while sitting on a rotating chair in front of an empty desk for safety. Participants interacted with substituted virtual objects of their real-world equivalent.

The demo application was developed in Unity 2020.1.17f1 [23] using the Magic Leap Unity template for that version provided on the official Magic Leap site, which includes the Magic Leap SDK. Sprites used inside the interfaces are part of Magic Leap's SDK or purchased sprite packs. The M3 lab provided material for manufacturing 3D objects, including tools, manufacturing products, cutting blueprints and mission descriptions.

8.2 Participants

Thirteen participants (two female) between the ages of 22 and 30 years old participated in the study. Six participants had nearsightedness, one astigmatism and one hyperopia. Participants were asked not to wear any contacts or glasses as they hampered the eye tracker's accuracy. We positioned the Glanceable interfaces closer to participants with nearsightedness until they stated they were clearly visible. Six participants had no previous experience in using AR. Two participants had bright eye colors, including cyan and green. Each participant was evaluated individually in order to comply with pandemic restrictions.

8.3 Procedure

The evaluation procedure consists of an object selection task presented in the form of manufacturing missions. Participants are given guidance using Holo box's 3-LOD system while they select the requested objects from the evaluation interface. Each mission requires selecting the correct tool, material and finished product from the evaluation interface and pressing a button to check their solution.

Initially, participants were asked to perform visual calibration for the Magic leap's eye tracker through the built-in calibration system in the device settings. Next, participants started the demo application through the device's main menu where they followed an initial tutorial which consists of four steps: (1) Interact with a Glanceable button via blinking, (2) Adjust the angle where Glanceable interfaces are visible at, (3) Place the evaluation interface and (4) Place the Holo-box interface. Placing the interfaces was achieved by placing the controller at the desired position and rotation and pressing a button. After the initial tutorial was completed, participants let go of the controller unless they wanted to re-position the interfaces at a later time.

8.4 Demos and Evaluation missions

After the initialization process, participants completed at least one demo training mission for machining. Participants could find the name of the mission and the correct tool, material and finished product through the 3-LOD interface. The task was to select the correct manufacturing tool, material and finished product from the EI. During demo missions, participants were free to take as much time and missions as necessary and make as many mistakes as they wanted; we did not record any metrics.

The main experiment consisted of four distinct missions, repeated in three epochs in randomized order. The first and second epochs consisted of all four missions, and the third epoch consisted of two of the four missions for ten total for the whole experiment. Participants' task was to complete all ten missions by making as few mistakes as possible. We instructed participants to memorize each mission's content as much as possible and complete each mission by using only Glanceable interfaces if possible. After completing the main experiment, quantitative metrics captured were saved under a unique ID. Participants completed a survey that included qualitative and quantitative questions regarding their experience using the same unique ID for future comparison.

8.5 Metrics

There were four different categories for metrics. Quantitative data were captured automatically on the Magic Leap One device during the experiment. In addition, participants filled out a survey that included the System Usability Scale (SUS) [75], the NASA Task Load Index (NASA TLX) [76] as well as user preference questions, which were associated with the quantitative data.

Quantitative data: For each participant, Magic Leap logged the following: (1) the number of tutorial missions completed, (2) blink-delay interaction events, (3) hand interactions events, (4) time to complete each mission, (5) the number of errors made for each mission and (6) the number of Holo-Box LODs used for each mission (3= all LODs, 2= only Glanceable, 1= only SG).

System Usability Scale (SUS): The SUS consists of a set of questions regarding the usability of our proposed system during the experiment. Answers are reported on a 7-point Likert scale (1= very low, 7= very high). The questions include grading (1) Learnability, (2) Efficiency, (3) Memorability, (4) Accuracy, (5) Satisfaction, (6) Intuitiveness, (7) Naturalness and (8) Fun of our proposed system.

NASA Task Load Index (NASA TLX): The NASA TLX covers topics regarding the user experience during the experiment's procedure. Like the the SUS all questions are answered in a 7-point Likert scale. The questions included (1) Mental Demand (How mentally demanding was the experiment?), (2) Physical Demand (How physically demanding was the experiment?), (3) Temporal Demand (How hurried or rushed was the pace of the experiment?), (4) Performance (How successful were you in completing your given task?), (5) Effort (How hard did you try to accomplish your level of performance?) and (6) Frustration (How insecure, discouraged, irritated, stressed and/or annoyed were you?).

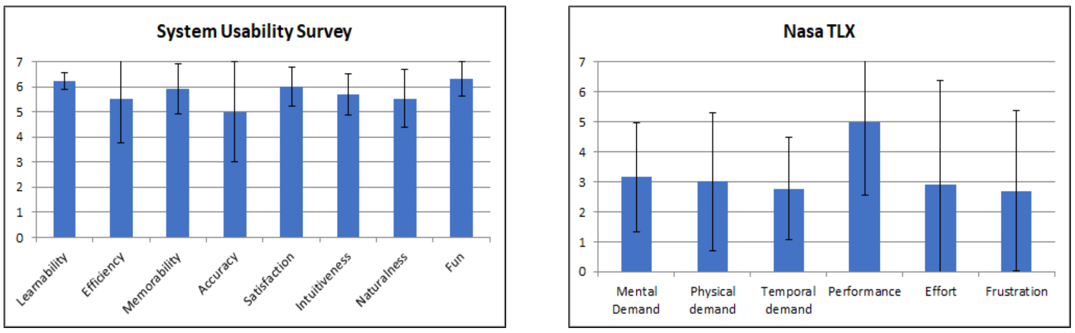
User Preferences: General questions regarding the two interaction types were administered. Participants were asked to select their preferred interaction method between blink-delay and hand-delay and then grade on a 7-point Likert scale (1) The accuracy of blink-delay interactions specifically on the two Glanceable interfaces SG and DG, (2) the precision of the eye-tracking cursor in regards to where they were looking and (3) the accuracy of the hand cursor in relation to their hand movements.

8.6 Hypotheses

These hypotheses guided our experiment:

- (H1) Participants will complete missions faster over time, with fewer errors.
- (H2) Participants will be able to complete subsequent missions while using only Glanceable interfaces.
- (H3) Our proposed blink-delay interactions can hide the Magic Leap's eye trackers' inaccuracies and users will not detect eye tracking failure, e.g., when the cursor freezes and does not follow the eye.
- (H4) Participants will prefer hand-delay interactions over blink-delay due to their higher accuracy.

8.7 Results



(a) (b)

Figure 118: (a) System Usability Scale (SUS) results (b) NASA Task Load Index (TLX) results

Mission completion metrics: When comparing the time completion for two participants on one mission, the times vary greatly based on individual skill. Instead, we calculated the average time improvement between epochs for each participant separately. We compare the average time on the first epoch with those of the following two. This way, we can understand if the participants improve over time regardless of their skills. For comparison, error rates and interface switches were similarly averaged per participant per epoch.

In the second epoch, participants completed missions with an average 24,16% (var: 9,6%) time improvement, while in the third epoch, participants showed a 39,04% (var: 13,38%) improvement over the first. Two participants showed no significant improvement (negative or below 10%) on all three epochs. In contrast, three more showed no significant improvement during the second epoch but showed significant improvement (35% and above) in the third.

When it comes to error rates, no significant improvement over time was shown with an average 0,46 (var: 0,45) errors per mission for the first epoch, 0,5 (var: 0,46) errors for the second and 0,46 (var: 0,82) errors for the third.

Regarding interface switches, participants used an average of 2,54 (var: 0,3) interfaces during the first epoch, 1,94 (var: 0,46) during the second and 1,77 (var: 0,53) during the third. The number of participants that used only Glanceable interfaces was one during the first epoch, four during the second and nine during the third. Six participants used all three interfaces on more than two out of four missions during the first epoch, two during the second epoch, while no participants used all interfaces on both missions of the third epoch.

In summary, H1 was partially supported as participants could complete missions faster over time, but the error rate showed no significant changes. H2 was supported as participants used fewer interfaces over subsequent missions and most missions during the second and third epochs were completed using only Glanceable interfaces.

Interaction preferences: When looking at the quantitative data, six participants preferred to use blink-delay interactions (over 60%), four hand-delay, and three participants used both interactions on an equal measure (both below 60%). Of the six participants who preferred blink-delay, five used blink-delay interactions (fewer than two hand-delay interactions) exclusively after the demo missions.

On the survey, eight participants stated they preferred blink-delay while five participants preferred hand-delay. When comparing the survey answers with the quantitative metrics of the same participant, only six participants preferred the same interaction type they mostly used during the experiment. Two out of five participants that used exclusively blink-delay stated they preferred hand-delay. Concerning the question "How accurate were the blink-delay interactions in relation to the Glanceable interfaces" the answers were positive, with an average of 5.46 (var: 3,17) on the 7-point Likert scale. Only two participants had significant difficulties performing blink-delay interactions due to a combination of eye tracking errors of the built-in tracker of the device as well as user-specific eye attributes. The participants used hand-delay instead.

Certain participants noted that while they preferred hand-delay inputs, they still used blink-delay inputs during the experiment. They found the novelty of eye-based inputs interesting for a short experiment without translating this to a preference for long-term usage. Two participants showed physical difficulties in keeping one eye closed while looking directly forward with the other to aim correctly, forcing them to keep their eyelids closed with their fingers to interact, eliminating the purpose of using blink-delay interactions. When grading the eye-tracking cursor's accuracy, the score was positive, with an average score of 4,54 (var: 1,17) and no scores below 3. Most participants stated that the blink-delay cursor was accurate and they did not notice inaccuracies in the cursor's position. Based on the supervisor's observations of the video feed and the participants' head movements, we noticed that in certain instances when the eye tracker could not detect the user's eyes and the cursor remained stationary, they instinctively rotated their neck to aim and interact correctly, resulting in higher perceived accuracy by the users. Conversely, when grading the hand tracking cursor, the average score was similar at 4.39 but with a significantly higher variance of 5.09. Most scores were above six or below 2, as the hand cursor's flickering was more apparent when the users' hands were stationary. All participants mentioned they noticed flickering on the hand cursor, with four participants not able to use hand-delay inputs.

In summary, even though the cursor in certain instances froze and moved away from interactable objects due to detection failure of the device, the overall perceived accuracy of blink-delay interactions was high. The only issue participants noticed was certain inaccuracies in detecting blinking. Thus, H3 was supported. On the other hand, we can neither confirm nor deny H4 with our results. With the combination of different preferences in interaction in the long and short term, interaction issues in relation to both interaction types for certain participants and the inability of two participants to physically use our blink-delay input, further research is required as well as improvement of gaze detection offered as part of current AR devices.

SUS: Our results on the SUS questionnaire are as follows (see Figure 118a): (1) Learnability avg = 6.23, var = 0.33 (2) Efficiency avg = 5.53, var = 1.78 (3) Memorability avg = 5.92, var = 0.99 (4) Accuracy avg = 5, var = 2 (5) Satisfaction avg = 6, var = 0.76 (6) Intuitiveness avg = 5.69, var = 0.82 (7) Naturalness avg = 5.53, var = 1.17 and (8) Fun avg = 6.3, var = 0.7. Based on these results we can see that Holo-box has positive scores in all fields, with (4) Accuracy having a lower score and higher variance than other values, further confirming the uncertainty in relation to H4 due to inaccuracies in both interaction systems.

NASA TLX: Similarly, our results from the NASA TLX questionnaire are as follows (see Figure 118b): (1) Mental Demand avg = 3.15, var = 1.82 (2) Physical Demand avg = 3, var = 2.30 (3) Temporal demand avg = 2.76, var = 1.71 (4) Performance avg = 5, var 2.46 (5) Effort avg = 2.92, var = 3.45 and (6) Frustration avg = 2.69, var 2.67. All results are one point off average, with performance being slightly positive. At the same time, demand, effort and frustration scores were slightly lower than average, showing that participants overall performed well with a low task load. The high variance in effort shows that certain users struggled significantly more than others while using Holo-box. This was confirmed through the discussion with the participants due to the occasional interaction inaccuracies of the device.

8.8 Conclusion

In this work we introduced Holo-Box, a novel interface system for information retrieval through Glanceable and 3D interfaces. Holo-Box uses a 3-LOD interface design that shows guidance information in progressing levels of detail as necessary. Interaction with virtual objects use blink-delay input for all interfaces, while 3D interfaces also use hand-delay input. These interaction types allow for precise interaction with virtual objects while limiting mid-air hand movements inside the bounding boxes of 3D interfaces for safety constraints. Interface content is adapted appropriately so that experienced users can complete assigned tasks using more compact interfaces, while inexperienced users can activate a denser LOD for additional guidance.

Evaluation of Holo-box occurred in a safe environment; users completed a selection task of required items in a manufacturing machine shop. Results showed that our system is easy to learn and users could complete tasks faster by using more compact interfaces over time. Both hand-delay and blink-delay interactions performed comparably well, with tracking issues of the device present in both. The blink-delay cursor was perceived to be accurate. Users could adjust their position using eye and head movements and they instinctively combined both activities when selecting objects. Our results show that our Holo-box AR interface is promising. The major drawback of our interface system were due to eye and hand tracking errors of the device. However, our choice of blink-delay interaction compensated for these errors to a degree.

8.9 Future work

Future work should compare our proposed interactions with other variations, such as hand-based interactions with better accuracy than Magic Leap One's stock tracking. Our proposed system could also be evaluated on the Magic Leap 2 AR HMD which will be released after the summer of 2022 and was not available during the course of our implementation. Similarly, Holo-box could be ported to other AR HMDs such as the Hololens 2.

Our 3-LOD interface system can also be used in other scenarios than manufacturing. Future work could check the viability of our interface system in different scenarios. Glanceable interfaces have been used in a plethora of scenarios including manufacturing tasks, collaboration tasks, social scenarios, virtual tours or augmenting everyday tasks such as cooking or drawing. Holo-box could be adapted to work under all of these conditions. Especially interesting are scenarios where the user could use more than one set of 3-LOD interfaces simultaneously for performing multiple tasks at once.

Another improvement could involve using custom hand or eye trackers with improved accuracy. In our work we focused on designing the 3-LOD interface system and we used the standard trackers integrated in the Magic Leap One, which proved to have coarse tracking. A more accurate tracking system would solve most of the issues reported during our evaluation.

One thing we omitted during our user study was measuring the effect of external parameters on the participants' performance. On our results, we stated that we expected our error rate to decrease while it remained static across the study. One possible explanation was that another factor, such as user fatigue, counteracts the positives of the learning experience.

Finally, future work could research different variations of our 3-LOD interface system. An example would involve using a different amount of interfaces other than 3. Another example could involve a more optimal presentation of the virtual interfaces to make them more immersive or visually appealing.

References

[1] Salva Kirakosian, Grigoris Daskalogrigorakis, Emmanuel Maravelakis, and Katerina Mania. Near-contact person-to-3d character dance training: Comparing ar and vr for interactive entertainment. In *2021 IEEE Conference on Games (CoG)*, pages 1–5. IEEE, 2021.

[2] Grigoris Daskalogrigorakis, Salva Kirakosian, Angelos Marinakis, Vaggelis Nikolidakis, Ioanna Pateraki, Aristomenis Antoniadis, and Katerina Mania. G-code machina: A serious game for g-code and cnc machine operation training. *IEEE EDUCON 2021 forthcoming, Games in Engineering Education Special Session*, 2021.

[3] Grigoris Daskalogrigorakis, Ann McNamara, and Katerina Mania. Holo-box: Level-of-detail glanceable interfaces for augmented reality. In *ACM SIGGRAPH 2021 Posters*, pages 1–2. 2021.

[4] George Chryssolouris, Dimitris Mavrikios, Dimitris Fragos, Vassiliki Karabatsou, and Kostas Pistiolis. A novel virtual experimentation approach to planning and training for manufacturing processes–the virtual machine shop. *International Journal of Computer Integrated Manufacturing*, 15(3):214–221, 2002.

[5] Wolfgang Kuhn. Digital factory-simulation enhancing the product and production engineering process. In *Proceedings of the 2006 winter simulation conference*, pages 1899–1906. IEEE, 2006.

[6] Frieder Loch, Mina Fahimipirehgalin, Julia N Czerniak, Alexander Mertens, Valeria Villani, Lorenzo Sabattini, Cesare Fantuzzi, and Birgit Vogel-Heuser. An adaptive virtual training system based on universal design. *IFAC-PapersOnLine*, 51(34):335–340, 2019.

[7] Ahm Shamsuzzoha, Rayko Toshev, Viet Vu Tuan, Timo Kankaanpaa, and Petri Helo. Digital factory–virtual reality environments for industrial training and maintenance. *Interactive learning environments*, 29(8):1339–1362, 2021.

[8] Zhiqiang Zhao, Da Jun Toh, Xiaoming Ding, Keng Chong Ng, See Choon Sin, and Yuen Choe Wong. Immersive gamification platform for manufacturing shopfloor training. 2020.

[9] Franziska Pürzel, Philipp Klimant, Volker Wittstock, and Michael Kuhl. Real nc control unit and virtual machine to improve operator training. *Procedia Computer Science*, 25:98 – 107, 2013. 2013 International Conference on Virtual and Augmented Reality in Education.

[10] George Alex Koulieris, Kaan Akşit, Michael Stengel, Rafał K Mantiuk, Katerina Mania, and Christian Richardt. Near-eye display and tracking technologies for virtual and augmented reality. In *Computer Graphics Forum*, volume 38, pages 493–519, 2019.

[11] Joseph Flynn, Steven Dance, and Dirk Schaefer. Industry 4.0 and its potential impact on employment demographics in the uk. *Advances in Transdisciplinary Engineering*, 6:239–244, 2017.

[12] Saliha Karadayi-Usta. An interpretive structural analysis for industry 4.0 adoption challenges. *IEEE Transactions on Engineering Management*, 67(3):973–978, 2019.

[13] Samat Imamov, Daniel Monzel, and Wallace S. Lages. Where to display? how interface position affects comfort and task switching time on glanceable interfaces. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 851–858, 2020.

[14] Feiyu Lu, Shakiba Davari, Lee Lisle, Yuan Li, and Doug A Bowman. Glanceable ar: Evaluating information access methods for head-worn augmented reality. In *2020 IEEE conference on virtual reality and 3D user interfaces (VR)*, pages 930–939. IEEE, 2020.

[15] William Buxton. Integrating the periphery and context: A new taxonomy of telematics. In *Proceedings of graphics interface*, volume 95, pages 239–246. Citeseer, 1995.

[16] Carlos Elmadjian and Carlos H Morimoto. Gazebar: Exploiting the midas touch in gaze interaction. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7, 2021.

[17] David Lindlbauer, Anna Maria Feit, and Otmar Hilliges. Context-aware online adaptation of mixed reality interfaces. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*, pages 147–160, 2019.

[18] Robert JK Jacob. What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 11–18, 1990.

[19] Sunwook Kim, Maury A Nussbaum, and Joseph L Gabbard. Augmented reality "smart glasses" in the workplace: industry perspectives and challenges for worker safety and health. *IIE transactions on occupational ergonomics and human factors*, 4(4):253–258, 2016.

[20] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994.

[21] Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385, 1997.

[22] Shohei Mori, Okan Erat, Wolfgang Broll, Hideo Saito, Dieter Schmalstieg, and Denis Kalkofen. Inpaintfusion: incremental rgb-d inpainting for 3d scenes. *IEEE Transactions on Visualization and Computer Graphics*, 26(10):2994–3007, 2020.

[23] Unity Technologies. Unity game engine.

[24] Epic Games. Unreal game engine.

[25] Wikipedia contributors. Duolingo — Wikipedia, the free encyclopedia, 2019. [Online; accessed 14-May-2019].

[26] Luis Filipe Rodrigues, Carlos J. Costa, and Abilio Oliveira. The adoption of gamification in e-banking. In *Proceedings of the 2013 International Conference on Information Systems and Design of Communication*, ISDOC '13, pages 47–55, New York, NY, USA, 2013. ACM.

[27] Markus Funk, Tilman Dingler, Jennifer Cooper, and Albrecht Schmidt. Stop helping me-i'm bored!: why assembly assistance needs to be adaptive. In *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pages 1269–1273. ACM, 2015.

[28] Annika Sabrina Schulz, Franziska Schulz, Rúben Gouveia, and Oliver Korn. Branded gamification in technical education. In *2018 10th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, pages 1–8. IEEE, 2018.

[29] Frieder Loch and Birgit Vogel-Heuser. A virtual training system for aging employees in machine operation. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 279–284. IEEE, 2017.

[30] Timothy James. Smart factories. *Engineering & technology*, 7(6):64–67, 2012.

[31] C. Gröger and C. Stach. The mobile manufacturing dashboard. In *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, pages 138–140, March 2014.

[32] G. Pritschow and S. Röck. "hardware in the loop" simulation of machine tools. *CIRP Annals*, 53(1):295 – 298, 2004.

[33] Xiang Yang, René C. Malak, Christian Lauer, Christian Weidig, Hans Hagen, Bernd Hamann, Jan C. Aurich, and Oliver Kreylos. Manufacturing system design with virtual factory tools. *International Journal of Computer Integrated Manufacturing*, 28(1):25–40, 2015.

[34] Néstor Ordaz, David Romero, Dominic Gorecky, and Héctor R. Siller. Serious games and virtual simulator for automotive manufacturing education and training. *Procedia Computer Science*, 75:267 – 274, 2015. 2015 International Conference Virtual and Augmented Reality in Education.

[35] Q.-H. Wang, J.-R. Li, and H.-Q. Gong. A cad-linked virtual assembly environment. *International Journal of Production Research*, 44(3):467–486, 2006.

[36] G.-C. Vosniakos, J. Deville, and E. Matsas. On immersive virtual environments for assessing human-driven assembly of large mechanical parts. *Procedia Manufacturing*, 11:1263 – 1270, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[37] Michael Schenk, Steffen Straßburger, and Heike Kissner. Combining virtual reality and assembly simulation for production planning and worker qualification. 09 2005.

[38] G. Michalos, S. Makris, N. Papakostas, D. Mourtzis, and G. Chryssolouris. Automotive assembly technologies review: challenges and outlook for a flexible and adaptive approach. *CIRP Journal of Manufacturing Science and Technology*, 2(2):81 – 91, 2010.

[39] Ann-Christine Falck, Roland Örtengren, and Dan Högberg. The impact of poor assembly ergonomics on product quality: A cost–benefit analysis in car manufacturing. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 20(1):24–41, 2010.

[40] Galina Ivanova, Yuksel Aliev, and Aleksandar Ivanov. Augmented reality. textbook for future blended education. In *Proceedings of the International Conference on e-Learning*, volume 14, pages 130–136, 2014.

[41] Mhd Wael Bazzaza, Buti Al Delail, M Jamal Zemerly, and Jason WP Ng. iarbook: An immersive augmented reality system for education. In *2014 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, pages 495–498. IEEE, 2014.

[42] Fotis Liarokapis. Using Activity Led Learning for Teaching Computer Graphics Principles Through Augmented Reality. In Jean-Jacques Bourdin and Amit Shesh, editors, *EG 2017 - Education Papers*. The Eurographics Association, 2017.

[43] P. Maier and G. Klinker. Augmented chemical reactions: An augmented reality tool to support chemistry teaching. In *2013 2nd Experiment International Conference (exp.at'13)*, pages 164–165, Sep. 2013.

[44] Carolina Mateo-Segura. A new augmented-reality platform for electromagnetic education. In *2018 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, pages 174–177, United States, 11 2018. IEEE.

[45] Marius Erdt, Jason Maroothynaden, Junming Peng, Wolfgang Müller-Wittig, and Paul Gagnon. Augmented Reality as a Tool to Deliver e-Learning based Blended Content in and out of the Class-room. In M. Bronstein and M. Teschner, editors, *EG 2015 - Education Papers*. The Eurographics Association, 2015.

[46] T. Blum, V. Kleeberger, C. Bichlmeier, and N. Navab. mirracle: An augmented reality magic mirror system for anatomy education. In *2012 IEEE Virtual Reality Workshops (VRW)*, pages 115–116, March 2012.

[47] M. W. Bazzaza, B. Al Delail, M. J. Zemerly, and J. W. P. Ng. iarbook: An immersive augmented reality system for education. In *2014 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, pages 495–498, Dec 2014.

[48] Jorge Martín-Gutiérrez, José Luís Saorín, Manuel Contero, Mariano Alcañiz, David C. Pérez-López, and Mario Ortega. Design and validation of an augmented book for spatial abilities development in engineering students. *Computers & Graphics*, 34(1):77 – 91, 2010.

[49] Yuksel Aliev, Vasil Kozov, Galina Ivanova, and Aleksandar Ivanov. 3d augmented reality software solution for mechanical engineering education. In *Proceedings of the 18th International Conference on Computer Systems and Technologies*, CompSysTech'17, pages 318–325, New York, NY, USA, 2017. ACM.

[50] A. Marinakis. Augmented reality for cad-cam training featuring 3d interactive geometric transformations. http://purl.tuc.gr/dl/dias/08EC772E-BE8A-4A6B-83E5-694BEC36C4B1, 2019. Accessed: 2019-04-11.

[51] Andrew Nee, S K Ong, George Chryssolouris, and Dimitris Mourtzis. Augmented reality applications in design and manufacturing. *CIRP Annals - Manufacturing Technology*, 61:657–679, 12 2012.

[52] Loukas Rentzos, Stergios Papanastasiou, Nikolaos Papakostas, and George Chryssolouris. Augmented reality for human-based assembly: Using product and process semantics. *IFAC Proceedings Volumes*, 46(15):98 – 101, 2013. 12th IFAC Symposium on Analysis, Design, and Evaluation of Human-Machine Systems.

[53] Gabyong Park and Woontack Woo. Hybrid 3d hand articulations tracking guided by classification and search space adaptation. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 57–69. IEEE, 2018.

[54] Xuan Wang, SK Ong, and Andrew Yeh-Ching Nee. Multi-modal augmented-reality assembly guidance based on bare-hand interface. *Advanced Engineering Informatics*, 30(3):406–421, 2016.

[55] Holo-board: an augmented reality application manager supporting machine vision. http://purl.tuc.gr/dl/dias/B6D12911-3767-46F9-B611-67F37486E4E6. Accessed: 2019-04-11.

[56] Anup Doshi, Shinko Yuanhsien Cheng, and Mohan Manubhai Trivedi. A novel active heads-up display for driver assistance. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):85–93, 2009.

[57] Yung-Ching Liu and Ming-Hui Wen. Comparison of head-up display (hud) vs. head-down display (hdd): driving performance of commercial vehicle operators in taiwan. *International Journal of Human-Computer Studies*, 61(5):679–697, 2004.

[58] Zeljko Medenica, Andrew L Kun, Tim Paek, and Oskar Palinko. Augmented reality vs. street views: a driving simulator study comparing two emerging navigation aids. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 265–274. ACM, 2011.

[59] Vassilis Charissis, Stylianos Papanastasiou, and George Vlachos. Comparative study of prototype automotive hud vs. hdd: collision avoidance simulation and results. Technical report, SAE Technical Paper, 2008.

[60] SeungJun Kim and Anind K Dey. Simulated augmented reality windshield display as a cognitive mapping aid for elder driver navigation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 133–142. ACM, 2009.

[61] Kyung S Park, Il Haeng Cho, Gi Beom Hong, Tek-Jin Nam, Jinyung Park, Seong Ik Cho, and In-hak Joo. Disposition of information entities and adequate level of information presentation in an in-car augmented reality navigation system. In *Symposium on Human Interface and the Management of Information*, pages 1098–1108. Springer, 2007.

[62] H Sawano and M Okada. Real-time video processing by a car-mounted camera and its application to car navigation systems by an augmented reality-based display method. *The Journal of the Society for Art and Science*, 5(2):57–68, 2006.

[63] Richie Jose, Gun A Lee, and Mark Billinghurst. A comparative study of simulated augmented reality displays for vehicle navigation. In *Proceedings of the 28th Australian Conference on Computer-Human Interaction*, pages 40–48. ACM, 2016.

[64] Roel Vertegaal et al. Attentive user interfaces. *Communications of the ACM*, 46(3):30–33, 2003.

[65] Ann McNamara, Chethna Kabeerdoss, and Conrad Egan. Mobile user interfaces based on attention. 2015.

[66] Ann McNamara, Katherine Boyd, Joanne George, Weston Jones, Somyung Oh, and Annie Suther. Information placement in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 1765–1769. IEEE, 2019.

[67] Shakiba Davari, Feiyu Lu, and Doug A Bowman. Occlusion management techniques for everyday glanceable ar interfaces. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 324–330. IEEE, 2020.

[68] J Zhu, Soh-Khim Ong, and Andrew YC Nee. A context-aware augmented reality assisted maintenance system. *International Journal of Computer Integrated Manufacturing*, 28(2):213–225, 2015.

[69] Feiyu Lu and Doug A Bowman. Evaluating the potential of glanceable ar interfaces for authentic everyday uses. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 768–777. IEEE, 2021.

[70] Shakiba Davari, Feiyu Lu, and Doug A Bowman. Validating the benefits of glanceable and context-aware augmented reality for everyday information access tasks. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 436–444. IEEE, 2022.

[71] Ken Pfeuffer, Yasmeen Abdrabou, Augusto Esteves, Radiah Rivu, Yomna Abdelrahman, Stefanie Meitner, Amr Saadi, and Florian Alt. Artention: A design space for gaze-adaptive user interfaces in augmented reality. *Computers & Graphics*, 95:1–12, 2021.

[72] Allison Jing, Kieran May, Gun Lee, and Mark Billinghurst. Eye see what you see: Exploring how bi-directional augmented reality gaze visualisation influences co-located symmetric collaboration. *Frontiers in Virtual Reality*, 2:79, 2021.

[73] Feiyu Lu, Shakiba Davari, and Doug Bowman. Exploration of techniques for rapid activation of glanceable information in head-worn augmented reality. In *Symposium on Spatial User Interaction*, pages 1–11, 2021.

[74] Magic leap: Enterprise augmenter reality (ar) platform. https://www.magicleap.com/en-us/. Accessed: 2022-06-03.

[75] James R Lewis. The system usability scale: past, present, and future. *International Journal of Human–Computer Interaction*, 34(7):577–590, 2018.

[76] Sandra G Hart. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 50, pages 904–908. Sage publications Sage CA: Los Angeles, CA, 2006.