UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

Volumetric display for a continuous viewing of a 3D object through projection

Tesi di:
Gabriel LOGRONIO,
matricola 930620

Relatore:
N. Alberto BORGHESE
Correlatore:
Renato MAINETTI

A.A. 2019/2020

Alla mia famiglia…
Grazie.

TABLE OF CONTENTS

ABSTRACT

Various devices on the market are tagged as volumetric display, none of them allow to display and then view a 3D object from any angle or point of view.

In this thesis we present how to realize a volumetric display that allows the user to observe a 3D object from any point of view around it.
To achieve this target, the device utilizes a Microsoft's Kinect to follow the user's position and calculate the view of the object from the given point of view and render it on a display.
The image that appears on the screen is then reflected onto a semitransparent and reflective cone made of plastic for the user to see.
The effect we achieved is the 3D object projected and hovering in middle air.

We then designed various interaction paradigms to interact with the displayed projection: using Amazon Alexa the user can interact vocally, the same Kinect used for tracking the user's position is used to interact through gestures.

We presented each step of the implementation of the volumetric display, with a particular focus on the research that was conducted regarding the methods to achieve each step, giving an explanation on why every technique was deemed more or less fitting for the use we wanted to make.

CHAPTER 1

Introduction

If I was to ask you to think of a hologram, it would be pretty easy to guess that the first image that comes into your mind would be the Princess Leia's famous holographic message ("Obi-Wan Kenoby, you're my only hope").
So is a hologram is device capable of displaying or projecting a virtual object for anyone to see and from any point of view without the need of any special equipment?
This conception is actually wrong, as holograms have a slightly different meaning from what we are used to: the correct term to refer to these devices is "volumetric display", as we will see a hologram is the 2D captured image of a 3D light map.

The goal of this thesis is to produce and develop a "true" volumetric display, that can be then used in different areas such as medicine, education or advertisement.
The design started from the idea of creating a device that allows multiple users to view the same 3D object with the need of a single display for all the users and without the need of specific tools or objects (i.e. glasses, controllers, ecc) for each user that acts as an intermediary.
The analysis of other techniques such as virtual reality, which require a special headset, and augmented reality, that can use device as glasses or smartphones, showed that our required solution should keep the user completely free to view and interact with the virtual object.
Many devices have been developed to reach this same goal but one way or another each of them failed in a requirement that we deemed essential, in particular they all presented a limitation in the points of view, ranging from a single point of view, to a small and limited set of points of view or a small angle of view, while our target is to reach a 360° continuous view all around the object, without limit on the user's possible positions and point of view.

Finally we tried to implement a method to interact with the device that felt fluid and natural, avoiding again to use of any sort of intermediary device that could feel like a bottleneck between the user and the display and therefore hinder the experience.

We started from the works of Xuan Luo, a student from the University of Washington, who developed an "Inexpensive DIY 3D Display"[1], and expanded her project to fulfill our ideas and reach our goal:
using devices such as the Microsoft's Kinect[1], the Amazon's Echo Dot and Amazon's Alexa[2] and the Unity game engine[3], our developed device will track the user's position, to show him the right point of view of the projected object, allowing him to view the said object from all around it, giving a 360° continuous point of view.

This paper has been divided in 8 parts: we will start by explaining the difference between a hologram and a volumetric display, as the two terms have been wrongfully confused, to then get a quick view of the current State of the Art of volumetric displays with a list of different techniques that each try to approach the problem with a different point of view and therefore each develop a different solution, with a focus on their limits.
We will then speak in detail about our solution, giving a view on different possible approaches to achieve this solution and how each of them compare: from the general setup of the components of the display and the connection between them, to the functioning principle of every aspect of the device.
This includes the tracking of the user's position using the Kinect, the use of this position to generate and display the image for the user to see through the reflection and the various interaction paradigms we developed to interact with the device and its "hologram".
Lastly we will see various applications of our device and other future works that might expand the possibilities of the device to thin the gap between reality and fiction.

[1]https://developer.microsoft.com/en-us/windows/kinect
[2]https://developer.amazon.com/en-US/alexa
[3]https://unity.com/

CHAPTER 2

State of Art

Since its invention in 1948, holograms have been subject of interest, both for research purposes and entertainment.
But contrary to popular belief a hologram is not the 3D display of a virtual object, as the correct term to refer to such a display is “volumetric display”.
But they are not completely different techniques as they aim the a similar goal: as the holograms are optical informations of a 3D light map stored into a 2D image for later retrieve and process, a volumetric display is the use of a 2D image that represents a 3D light map to mimic and display the same 3D light map.

2.1 Holograms

A Hologram is an optic technique that allows to record a visual information, like a 3D object or a more complex 3D scene, as a three-dimensional light field and store it as an image in the form of an interference pattern, that retains the original properties such as depth and parallax, onto a holographic plate [1][2][3].
The created image is the so called hologram.

The hologram, in the right environment, i.e. light, can “reproduce” the original light field. Since it maintains the parallax and perspective, the hologram can be viewed from different angles, therefore a hologram is in effect a three-dimensional image and no just an image with depth.
Holograms have found different uses: from the 1960s and 70s it entered in art exhibitions, in 2005 it was first utilized to produce data storage medium, but as today they are mostly used in security, like in many currencies and credit cards (in the **Fig 2.1** we can see the hologram used on the 10 euro paper bill) [1].

Fig 2.1 Holograms on a banknote

In the **Fig 2.2** we can see the operating principle of the image recording of a hologram: a laser beam emits a monochromatic (therefore with a single wavelength) and coherent (with all the waves in phase) light.

The emitted laser is splitted up by a semi-transparent mirror in two rays: an object beam and a reference beam.
The first one is used, through mirrors or lenses that direct the light, to enlighten the object, that has to be completely still for he whole recording session, while the other is directly channeled with the lenses to the holographic plate of a photo-sensitive like photopolymers, onto this plate the two beams collide and interact I a way that is called "wave interference": the two waves of the beams superpose creating a wave which amplitude is the sum of the amplitudes of the beams.
The holographic plate is set to register and store the resulted wave as the final hologram [1][3].

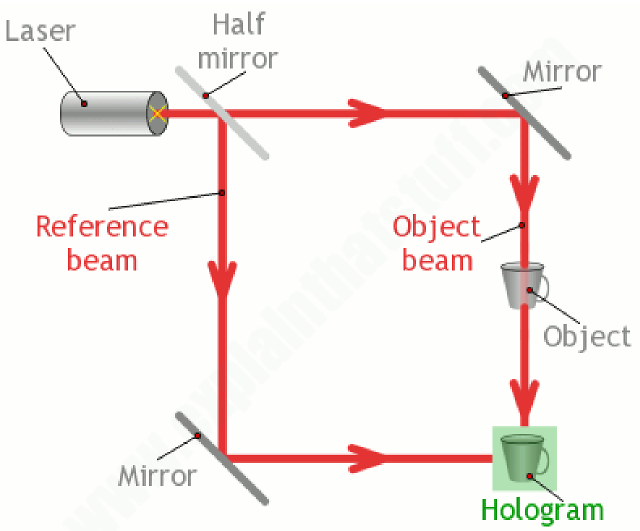


Fig 2.2 From 3D to 2D: how to create a hologram

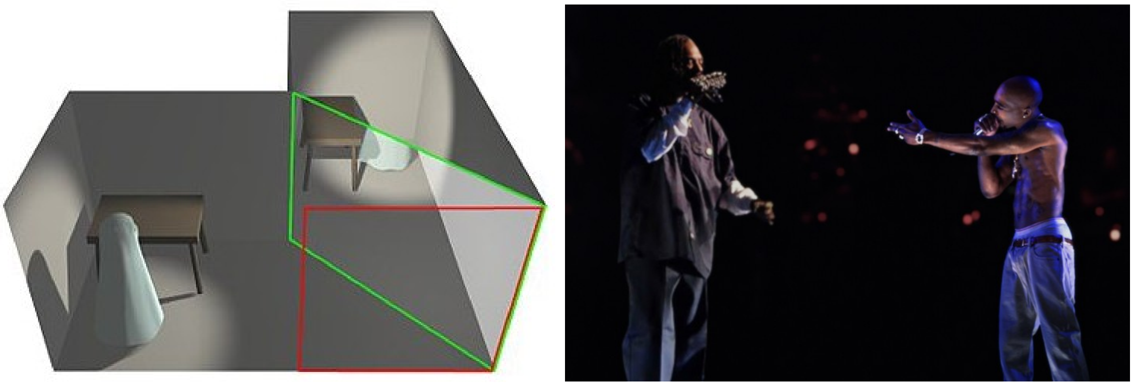
2.2 2D Image projectors

These devices mimic volumetric displays by using a 2D image that simulate depth through visual effects.
The issue is common to all the devices in this category, as they have only one fixed point of view, directly in front of the display.
If the position requirement is fulfilled tho the effect is quite good and can trick the eye.

2.2.1 Pepper's Ghost

The Pepper's ghost is an illusion technique, invented by John Henry Pepper, mostly used in theater, and other entertainment medium.
While not being a volumetric display per se, it's
As we can see in **Fig 2.3a** the setup consists of 2 different scenes: the first scene is seen through the glass by the audience, while the second scene is reflected towards the audience through a beam splitter (in the form of a flat glass sheet) similar to the hologram, but for the opposite purpose: it combines the two images. The scene lighting is set to selectively illuminate given parts of the scenes, keeping the glass invisible to the audience.
Modern Pepper's ghost are made with a display that replaces the reflected scene in order to expand the possibilities of the virtual objects that can be added to the real scene, as in **Fig 2.3b** where a real person is next to the representation of a person passed away [4][5].

The issue: as per all the devices in the category, this method allows only one point of view given by the orientation and position of the glass panel.
It is in fact not a reconstruction of a 3D scene but it's just the overlapping of a 2D image onto an existing 3D scene.



(a) An explanation of the effect (b) A famous use of the technique

Fig 2.3 Pepper's ghost effect

2.2.2 Holographic fan

The holographic fan consists of 2 or 4 spinning rays with LED lights.
While spinning, a software controls the lights turning them on and off and changing the colors.
The result is the projection of an image that seems to float in the air, because of the rotational speed of the rays make them practically invisible to naked eye, (**Fig 2.4**), [5].
It can be used alone or in a group to create a bigger image.
This technique is often used in marketing and in stores.

The issue: the fan has to be installed onto a wall, which is both an additional issue (the structure is a constraint) and a solution for the main problem, as it forces the user to view the device from the front.

Fig 2.4 An example of a holographic fan

2.3 Volumetric displays

A volumetric display is a graphic display device that produces a 3D representation of an object or scene, as opposed to the previously seen techniques.
Volumetric displays are autostereoscopic, as they don't need particular objects such as glasses or smartphones and can bee view by unaided eye [6].
These displays are what science-fiction and we have wrongfully referred to as "holograms", an example is the classic holo-message from Star Wars (**Fig 2.3**).
Contrary to the holograms, there isn't an official process to produce a volumetric display, as there are different ways to develop a volumetric display device [7].
Therefore there are various different devices and each take a different approach to the design and develop of display: from different displaying technologies, to different use cases.

We will now present some of the volumetric displays currently present on the market, pointing at their flaws that our device is aimed to overcome [4][7][8].

2.3.1 Voxon VX1

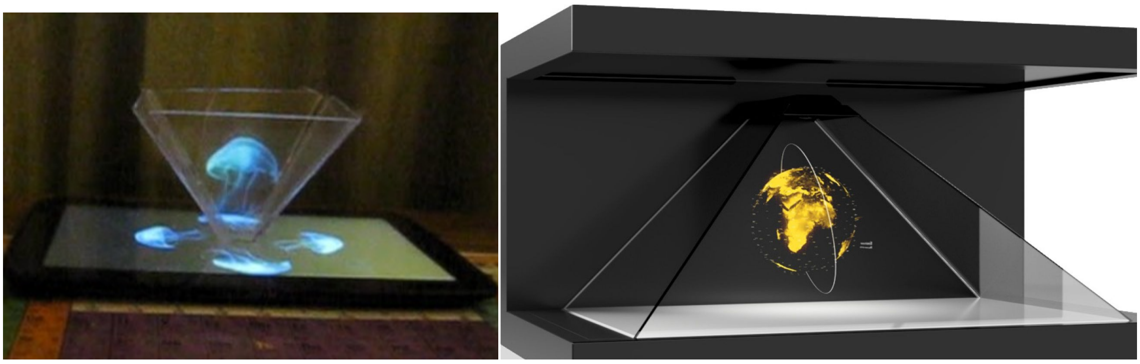
The VX1 is the best example of a volumetric display present on the market.
The best way to describe its functioning is "3D printing in the air": the virtual 3D object to be projected is sliced in hundreds of horizontals layers.
Each of these slices is projected in turn on a rear-projecting glass that moves up and down at 15 cycles per second giving the impression of continuity.
The result is a projected 3D can be viewed by any point of view.

The issue: the device has a high cost of around 10.000 $.
The mechanical part do wear out through time (around 500 hours of using are expected) and then must be changed, this raises the maintenance cost.
The area of projections is quite limited (18x 18 x 8 cm) and it's not scalable as the plate can not move fast enough between bigger distances to give the illusion of continuity.

2.3.2 Holographic pyramids

This system applies the Pepper's ghost illusion to create a volumetric display, but it uses a screen or a projector instead of a second scene.
Four glass panels are arranged to shape a pyramid that is then placed on top of the display, for the reflection to to be viewed by the user.
There are two variants that apply the same principle:
- the DIY pyramids, like the one in **Fig 2.5b**, are usually used for smartphones and tablets for the "holograms at home" effect, being quite cheap but inaccurate,
- the professionals pyramids pre-built within a given cubic structure, containing both the pyramid and a screen/projector (**Fig 2.5a**) [5].

Problem: while this technique allows 4 different points of view, compared to the only 1 of the Pepper's Ghost, it still limits the position of the user to only 4 sides of the pyramid, while the edges of the pyramid strongly separate the views, preventing the continuity and link between the views.



(a) DIY pyramid with a smartphone (b) A full system pyramid
Fig 2.5 Two examples of pyramid holograms

2.3.3 Looking glass:

The Looking Glass[1] is a glass box that manages to display a 3D object, with 45 different possible points of views over a 45°/50° wide cone in front of the glass. To each view is linked a 2D image, based on the position of the user, a different image is shown to the user. An example can be seen in **Fig 2.5b**. This way both the parallax and the perspective is maintained.
To make it happen it combines a light-field and a volumetric display, [5][9].

The issue: the looking glass has a preset set of different points of view that give a clear and almost continuous view of the object from different views, but it only covers a 45° cone in front of the display.

(a) A butterfly made of floating points (b) Two examples of Looking Glass

Fig 2.6 Two different volumetric displays

2.3.4 Ionizing laser beam

This particular laser beam utilizes femtosecond bursts to ionize the air into plasma, creating floating particles that can be moved into position to create a floating object (in **Fig 2.5a** we can see a butterfly).
The technology is still on the working and its precise functioning is still in the developing therefore more informations cannot be given at the moment, but it’s the most advanced technique and it’s the nearest research has reached to a pure volumetric display without limitations [5].

The problem: the technology is still in its early stages of development and utilizes high technology laser beam. Its technology is not available to the public. The projection is a 3D object visible from any point view, reaching the goal. Unfortunately the laser beam cannot change its color, making it hard to create a dynamic object.

[1]https://lookingglassfactory.com/

2.3.5 Pepper’s Cone

The Pepper’s Cone Project, by Xuan Luo, is a 3D Display that resumes from the holographic pyramid and aims to achieve a 360° continuous view [10].
A cone is used instead of a pyramid in order to overcome the problem of the 4 seams that limit the continuity of the vision. The image is pre-distorted, in order to appear straightened once reflected on the cone.
The screen used to create the reflection is provided by a smartphone or a tablet, and the same device is used for its gyroscope: rotating the device permits to rotate the object itself, allowing to view different sides of the 3D object [10].

The issue: the hologram has a single point of view, rotating the device allows to view different sides of the object, but the position of the user is still strictly fixed in front of the cone.

CHAPTER 3

The Setup

The 3D display consists of 4 different parts that, alongside the computer, allow to track the position of the user, create the hologram and interact with it (**Fig 3.1**).
The Kinect is placed as high as 1.80m and parallel to the ground (neither tilted forwards nor backwards) facing a wide area of roughly 6m^2.
In the middle of this area and at least 2m from the Kinect a screen is set horizontally facing upwards. On top of it the reflective cone is placed, point up.
Under the screen is placed the Echo Dot, to have to audio and the visual output coming from the same place and therefore make the projection consistent.
If the Echo Dot is not used then the audio output connected to the computer should replace the device in the same position.
The Kinect and the screen must be connected to the computer.
The computer and the Echo Dot must be connected to the internet.

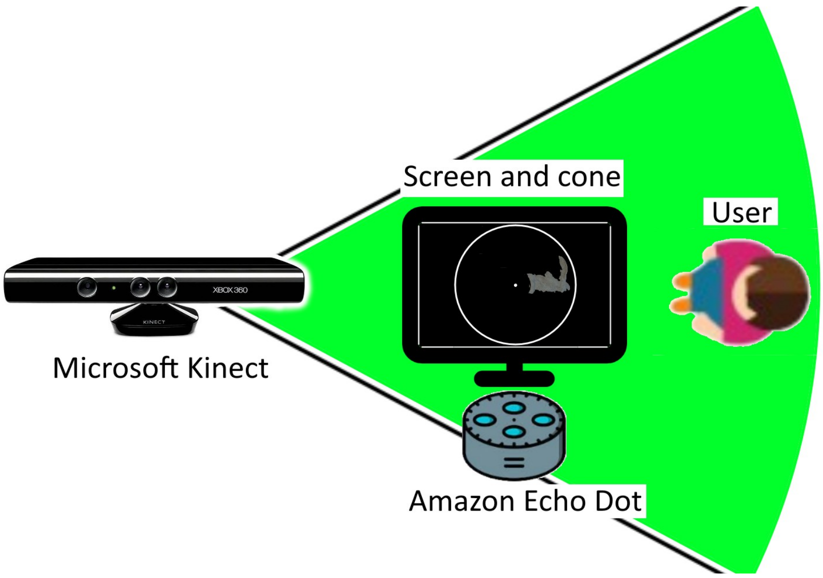


Fig 3.1 The setup from top view

3.1 The hardware

3.1.1 The screen

The screen provides the image that will reflect on the cone to then be viewed by the user.
It has a solely output purpose and therefore it doesn't have particularly demanding requirements:

- the dimension of the screen is strictly related to the cone dimension, and has a wide range of options, going from small 5' smartphones to a bigger 30' inch television and more, but it's important to keep a reasonable screen size/resolution ratio to have a smooth image without visible pixels.

- the resolution should be ah high as possible, but even a lower resolution works just as fine, and it is obviously related to the screen dimension.
We used a 15.6' inch, 1920x1080, 141 dpi laptop and a 30' inch, 1440x900, 56 dpi television, with satisfying results with both the screens.

- the refresh rate and fps also have low requirements. In modern screens 60Hz are the minimum standard while the fps are related to the computer: 30 fps are considered the minimum to "trick" the eye into a sense of continuous motion.

- the brightness of the screen plays a big role in the result of the reflection.
Because of our device utilizes the reflection of a light source onto a transparent material, it is most suitable in a dark environment.
A screen with high brightness is preferable, goingfrom 200 (the minimum standard in today screens) nits to up at 500 nits.(used in more premium and high end screens).
Our testing screen had a 200 nits maximum brightness.

- the problem of the backlight bleed and deep black color: to give a better illusion of a floating object, the part of the image not containing the object should be as dark as possible, which is not always possible because of color inaccuracy and backlight bleeding: all or part of the pixels are lighter and therefore can be seen on the reflection.
This is a problem strictly related to the screen and that is beyond our means and control and that will therefore be ignored.

3.1.2 The reflective cone

The reflective cone is the actual display of the hologram, has it's the medium through which the object can be seen.
As demonstrated by Xuan Luo in her resarch [10], the ideal angle of the cone for the best reflection is around 51°, but we decided to use a 45° cone, close enough to the suggested value for the reflection to be clearly visible and not distorted.

This decision was taken for two main reasons:
First of all this way the reflected visual rays are parallel to the screen itself, hence the height of vision of the user is independent of the distance from the screen: as per **Fig 3.2** the further the users in blue and red are from the screen, the higher or lower respectively they must be compared to the cone. Instead the user in black can maintain his height independently from the distance.
Secondly a 45° angle allows to create a neither too flat nor too high cone, as the height is half the length of the diameter, which is chosen as the shortest side of the screen in order to fill the space without excess.

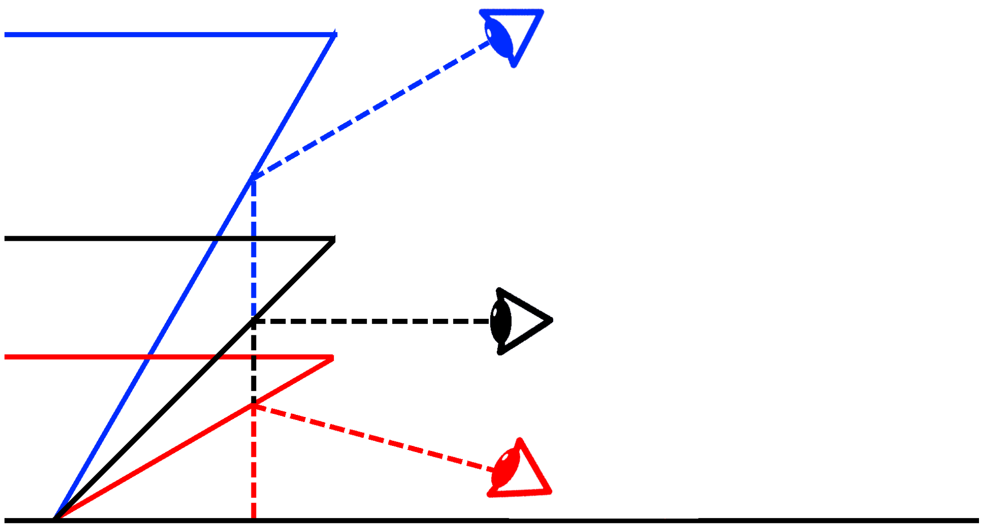
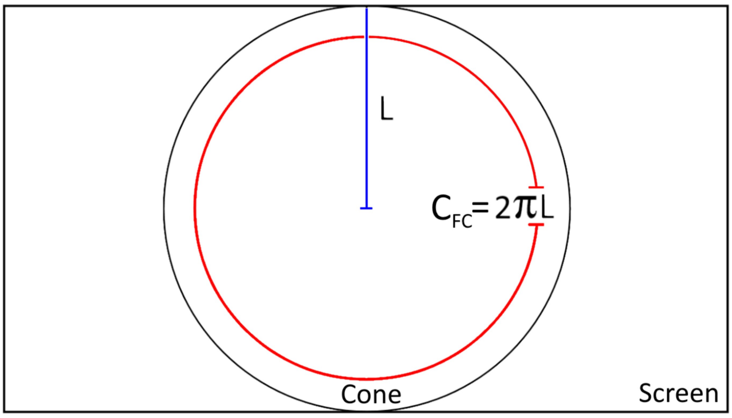


Fig 3.2 Different slopes of the cone

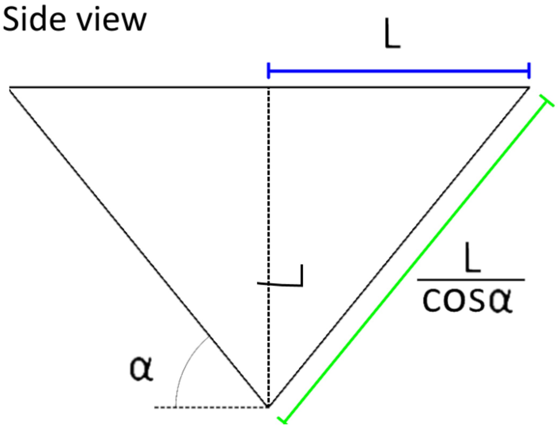
To build the cone we have to utilize 2 important measures: the angle of the cone (α) and the length of the shortest side of the screen (L).
We will refer to the final built cone as the FC ("Final Cone"), and the printed disk to be cut as CD ("Construction Disk").
Given this two values we can construct the cone as follows:

Top view



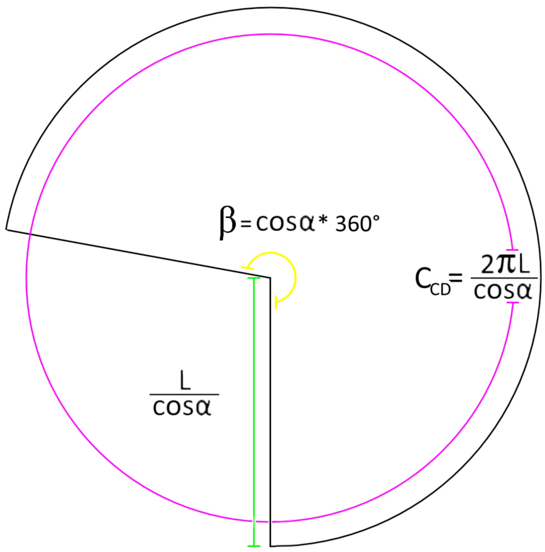
Given a generic 16:9 display, only a part of it is used for the reflection: the central LxL square, where the reflection will rotate around the middle of the screen.
The FC will be placed onto this central zone.

Side view



Knowing the length L and the angle α (that we set to 45°), we can calculate the length of the side of the FINAL CONE, that will be the radius of the CD.

Cone construction



To calculate the center angle we apply the ratio between arc of circumference and center angle

$$\frac{C_{FC}}{C_{CD}} = \frac{\beta}{360°}$$

We can build the cone from a circle with a radius of $R = L/\cos\alpha$ and a center angle of $\beta = \cos\alpha * 360°$ (in our case $\approx 255°$).
Print the shape onto a plastic sheet and cut it out, then proceed by attaching the two sides of the center angle to obtain the final cone.

The material used to build the cone is a PET plastic sheet, 0.25cm thick, but as demonstrated in the original work, any reflective and transparent material neither too thick nor too thin can be used.

3.1.3 The Microsoft Kinect

The Kinect device is used both for tracking the user position and to interact with the hologram. The Kinect is series of motion detection and tracking devices produced by Microsoft and first released in 2010 for the Xbox console, and then expanded for the Windows OS in 2012 [11].
It is based on human body recognition and skeleton extraction to identify the joints, then it applies position tracking and prediction of the joints to trace the movements.
We will see in detail in chapter 4 how the Kinect devices work.

3.1.4 The Amazon Echo Dot

The Amazon Echo Dot and its virtual personal assistant AI Amazon Alexa are used to interact vocally with the hologram, giving out the instructions and listening to the responses.
The Echo Dot is a smart speaker of the Amazon Echo lineup (**Fig 3.3**), developed by Amazon and firstly released in 2016, with a new version released every year. The device can provide a series of services: from weather and news, to music and radio. Developers can add new personalized programs, called “skills”, built with the Alexa Skills set and written mainly in Javascript [12].
We will see in detail in chapter 6 how the Echo Dot device works.

3.2 The Unity game scene

All the parts are put together in Unity, a game engine first developed in 2005. As the Kinect and the screen are directly connected to the computer, the Alexa connection is managed through internet connection, therefore both the components must be connected to the internet.

The scene contains different parts, the most important of which are:
- the Alexa Manager: manages the Unity-side of the connection Unity-Alexa, as we will see in chapter 6.
It receives messages sent from the Amazon Echo Dot through the PubNub server and decrypts them in order to extract the request and act accordingly, i.e. sending an answer message back, changing the object to display and more.

- the Kinect Manager: manages the tracking of the key body parts of the user: the head and the hands, as per chapter 4.
Whenever a user is tracked by the Kinect, a new prefab is spawned and the tracking begins. At each frame it checks whether the same user is still in its field of view to update the position of the joints.
If the user exits said area it despawns the prefab.
The hands movement can prompt different commands, to which the Kinect Manager reacts accordingly.
The head movements are followed to display the right view of the object.

- the Catalogue Manager: manages the catalogue and the 3D objects contained. Receives the orders from either Alex Manager or the Kinect Manager, to display a given object or a given detail of an object and browse between all the objects in the catalogue.
If requested it can close all models and display the virtual assistant.

- the Rotation Manager: retrieves the position of the user from the Kinect Manager and applies it to the object’s rotation.
It rotates Catalogue Manager to show the camera the correct point of view of the currently displayed object or assistant, and then rotates the image captured by the camera around the center of the screen to follow the user’s movement.

3.3 Instructions to run the device

To test the display, follow this instructions as per chapter 3.1, the steps marked by (Alexa) are needed only if the Alexa voice interaction pattern is selected:

1. Connect the Kinect to the PC via USB (USB 2.0 port for a Kinect for 360, USB 3.0 port for Kinect One) and to a socket for the power.

(Alexa) Connect the PC to the Internet: a fast connection is preferable to minimize the interactions delay. We suggest to connect the PC to a socket.
Connect the Amazon Echo Dot to the Internet: a fast connection is preferable to minimize the interactions delay.
(optional) Set the Amazon Echo Dot language.

2.. Connect a screen to the PC and place it horizontally in front of the Kinect: it's important that the screen is in the middle of the Kinect's field of view to ensure a correct tracing.
Mark the distance between the screen and the Kinect as it will be needed later. This step is important as the Kinect cannot track the screen position's so we have to put manually the position of the screen into the game scene.

3. Put the cone over the center of the screen.

4. launch the program "Pepper's Virtual Assistant".

5. insert the distance between the screen and the Kinect when requested.

6. Select the interaction paradigm
(Alexa) Launch the "Pepper's Virtual Assistant" skill from the Amazon Echo Dot.
Select the language.

7. Start the presentation from the beginning or skip directly to a given object.

CHAPTER 4

Tracking the user position

The user's position, and in particular his head, around the screen is tracked to ensure to show the user the right view of the object with continuity.

4.1 Other approaches

There are several methods to achieve it and we explored them to find the method that better suited the following requirements:
- a smooth tracking without snaps of the position,
- a good and stable refresh rate for a continuous tracking (30 Hz),
- a wide operating area (5 meters x 5 meters),
- no devices used as a medium by the user,
- reliable and subject to low noise
- rare to no loss of tracking.

4.1.1 Visual markers

A marker is a well recognizable image (or object) that can be easily detected and tracked by a device equipped with a camera (i.e. QR codes).
The camera frames the marker and through an inverse transformation it can extract its position relatively to the marker with high precision [13][14].
A marker would be placed near the display and the user would have to frame it with a device equipped with a camera and run an app especially developed.
The tracking is fast and reliable, with a generally large operating area assuming the marker always visible (no obstacles between the user and the marker).

The issue: requires a device to view the display through, a medium that interposes between the user and the screen.

4.1.2 Computer vision

This technique is the opposite of the previous method, as requires a simple RGB camera, preferably in high definition, that records and streams the scene to a computer [15].

The image is then processed through one of the following algorithms to extract the user's position (in **Fig 4.1** we can see an example applied in a street):
- A reference image is created beforehand by recording the empty setup (i.e. the only screen without and user) and calculating the average pixel by pixel to eliminate the noise.
At each frame the captured image is confronted with the reference image to spot the differences: through image cleaning algorithms the user is separated from the background and the position is set as the center of mass.
But the reference image should be recomputed every time the scene undergoes the slightest change (i.e. the user places a bag on the floor, the algorithm recognizes it as a user).
- A more advanced approach involves calculating a reference image based on the last N frames, making it dynamic to changes.
It might prove useful to try to predict the movement of the user tho smooth the movement, i.e. assuming uniform motion.
The approach is fast and reliable, with libraries already set to apply these algorithms,

The issue: it cannot precisely recogniz each body part of the user and even a simple hand movement can be perceived as a user movement [15].

Fig 4.1 The tracking the movement of people

4.1.3 Pressure sensors and Wii Balance Board

The Wii Balance Board[1] is a device released by Nintendo in 2007 for the Wii console (an example in **Fig 4.2**).
It is used to track the user's center of balance for both videogames and the physical fitness industry [16][17].

Through the four pressure sensor placed at the corners of the device, it can determine both the position of the user's center of balance and the user's weight.
By placing one, or more than one, underneath the floor around the screen, it can be used to detect the user's position relatively to the balance board center.
The device is so accurate that is it also used in the medical and fitness industry to assess the center of pressure of a person, in order to improve his balance and correct posture errors[16].
The WBB was reported to have three different acquisition rates (≈100 Hz, ≈54 Hz and ≈35Hz) with an average acquisition rate of 63 Hz, suited for the use we want to make of it [17].

The issue: the device's tracking is only reliable in the case of the user having both feet set on the ground, as when the user is moving the tracking "snaps" between the moment one foot gets up and the same gets back down on the ground, opposed to the fluid and continuous movement of other approaches.

Fig 4.2 The Wii Balance Board

[1]https://www.nintendo.com/wiifit

4.1.4 WiFi / Bluetooth signal

Most smartphones are equipped with the hardware that enables to connect to radio signals such as WiFi and Bluetooth and can therefore be utilized to track the device's position [18][19].

There are various techniques that utilize this signal to calculate the device's position, given a series of Access Points (i.e. a WiFi router with multiple antennas or Bluetooth beacons) with known position and known maximum signal strength, that the device is connected to:
- multilateration:
The device is connected to all the APs in its range.
It then calculates the distance from each AP by using either

RSSI the signal strength: depends on the distance from the relative AP,
RTT the round-trip time: depends on time it takes to send a data packet to an AP and receive an answer from the same AP depends on the distance from the given AP.

Obtained a range of possible distances from each AP, the device can extract roughly its position through triangulation.
This technique is fast but not accurate enough [13].

- fingerprinting: to each position within the APs range is associated a characteristic and unique array of key-value pairs, that consists of each AP ID code and relative RSSI, called fingerprint.
Each of this fingerprint is paired with its position and stored in a database.
While the device moves in the area, it compares its current fingerprint with the list of fingerprints in the database and extracts the most likely position through a weighted average of the most similar fingerprints.
This technique relays heavily on the discretization step used during the setup (while populating the database): if the step is too big, the tracking is too imprecise, while if it too small it slows the process both in the setup and in the query of the database, lowering the position update rate [13].

The issue: these techniques are too inaccurate for the use we want to make of them.
It would be necessary to access the user's smartphone or provide a device with the proper application installed.

4.2 The Microsoft Kinect

The Kinect allows to track the user’s position relative to the sensor itself.

At the time of writing 2 versions have been released: the Kinect v1 for the Xbox 360 in 2010 and the Kinect v2 for the Xbox One in 2014, while a non-gaming “successor” named Kinect Azure has been released in 2019 for developers only.
Since 2011 the sensors have been used in non-gaming environments as a depth sensor and body tracking device, especially in academia and researches.
Both can track the position of the joints of a human body but have many differences, from the operating principle to the hardware [20]:

(a) Kinect 360 (b) Kinect ONE

Fig 4.3 The two Kinects

	Kinect v1 (360)	Kinect v2 (ONE)
Picture	**Fig 4.3a**	**Fig 4.3b**
Main hardware components	- infrared projector - infrared camera - RGB Camera - 4 Microphones	- RGB Camera - infrared camera - Time of Flight technology - 4 microphones
Max users	2	6
Joints n°	20	25
FOV	57° horizontal x 43° vertical	70° horizontal x 60° vertical
Other differences	- is reliable since the boot of the device - accuracy is affected by distance and artificial lights	- needs to “pre-heat” before achieving reliable results - doesn’t have problems with distance and illumination

The two sensors differ in the depth map acquisition procedure:

Kinect 360: the structured light.
The Kinect projects an invisible and irregular light pattern that illuminates the scene. The infrared sensor detects the lights that bounces onto the objects in the scene and captures them.
For every pixel in the camera the Kinect compares the known light pattern with the data obtained from the pattern projected but distorted because of the scene, then can calculate the depth map [20][21].

Kinect ONE: time of flight.
The Kinect projects modulated infra red light and the sensor captures the rays that bounce on the scene.
The Kinect measures the time-of-flight of the infrared ray: closer point will have shorter time-of-flight, while farther point will have longer time-of-flight.
For every pixel in the camera it then calculates how the time-of-flight has effected and distorted the modulated projected pattern to calculate the depth map [20].

After obtaining the depth map, both apply edge detection to separate the background (non important objects) from the foreground (important objects), they try then to identify any moving objects, assuming that only people are moving in the scene.
With the aid of a highly trained AI the devices segment the moving objects into shapes (the body parts) and discard the object that do not fit the criteria and are not recognized as human bodies [21].

Code snippet 4.1
KinectManager.GetJointPosition:
Given the user ID (a number that uniquely identifies each user), and the bone ID (a number that uniquely identifies each joint of the tracked body of a user), the Kinect returns a Vector3 that indicates the position of the joint in the coordinates of the sensor.

Code snippet 4.1

```
Vector3 posJoint = KinectManager.GetJointPosition(userID, boneID);
```

The Kinect v2 is generally the better option, as it’s way more accurate, can track more body simultaneously and for each body can track more joints.
But due to the situation of 2020, this project has been developed using a Kinect v1.

The Kinect devices meet all the requirements needed for the project: the refresh rate is high and the tracking is smooth.
The operating area is quite large and its presence can go unnoticed by the user.
It doesn’t require any device for each user.
Has low noise and low tracking loss.

The devices have some flaws that might impact negatively the functioning of the device:
- the Kinect utilizes a low resolution depth camera that fails to recognize the small details such as the user’s face features. Therefore the device fails to recognize the front from the back of the user.
- the cameras don’t work with occlusions and a user hidden to the view cannot be tracked, for example in case of the screen standing in the way between the Kinect and the user. It can predict the current position of a hidden joint by the history of the positions, but generally with low accuracy.
If the user is standing side-facing, the farthest hand is tracked with low accuracy or not tracked at all.
- the device have a limited area in which it can recognize the human bodies, more than 1 Kinect might be needed in case of bigger screen and hologram.

Despite these problems, with a controlled setup we can work around them:
- it is safe to assume that the user is facing directly the display during all the session, to observe and interact with it, it wouldn’t be possible, and therefore would not make sense, to utilize it while facing any other way,
- the setup expects only the screen, the cone and the user in front of the Kinect and there shouldn’t be any object that can constantly and fully occlude the user,
The Kinect is placed at around 1.8m (see in chapter 3) and the screen is well below this level. The screen might cover the legs of the user but both the hands and the head should be easily in view.
- while being heavily limited, the range of the Kinect still provides at least a $6m^2$ of operating area which is enough for almost any use case.

4.3 The rotation

Using the Kinect e track the position of the head of the user, we must then calculate the rotation to apply to the virtual object in the Catalogue so the display shows the right point of view of the said object to the camera.

Given the following assumptions we can simplify the calculus of the rotation:
- the screen and cone have a known and fixed position throughout the session (see chapter 3).
The Kinect cannot track the screen’s position but it is needed to calculate the position of the user relative to the screen and extract the rotation.
- the scene’s reference system is set to overlap the Kinect sensor’s reference system in order to simplify the calculus.
- given the shape of the cone (see chapter 3), the height of the point of view of the user is fixed and can be therefore ignored, making the tracking 2D.
With these informations we can extract the rotation to give to the object.

4.3.1 Rotation through Unity instructions

A simple approach takes advantage of Unity’s and other game engine’s high level instructions and methods such as “transform.LookAt”: this method is called from a script attached to a gameobject and receives as a parameter a trimdimensional vector that represents the position of another gameobject. The result is the rotation of the gameobject in order to face directly the given position.

The rotation is automatically applied and can be easily extracted from the transform of the gameobject.
Code snippet 4.2
The simple instructions can rotate the object to face the user’s position, the final rotation is the rotation of the object around its vertical axis.

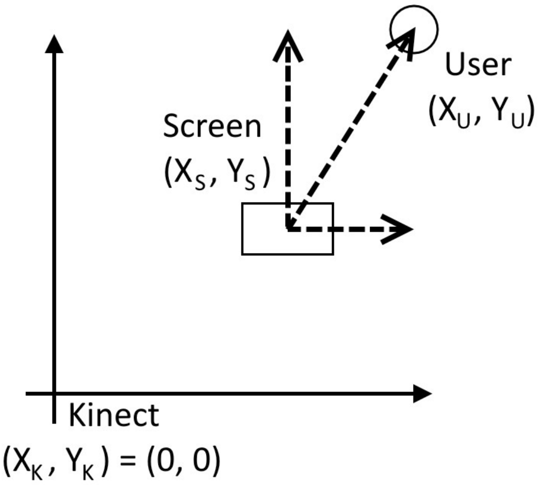
Code snippet 4.2

```
transform.LookAt(UserPosition);
float directionAngle = transform.localEulerAngles.y;
```

4.3.2 Rotation through trigonometry

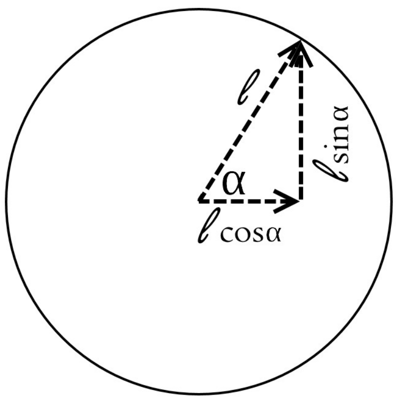
This approach can be applied to any development environment as it doesn’t use high level instructions to compute the rotation and it’s instead calculated.
To reach this goal we can follow these steps:
- calculate the direction of the vector screen-user
- calculate the euler angle of the direction vector

We can calculate the vector that indicates the direction of view simply by doing the subtraction of the two location vectors:

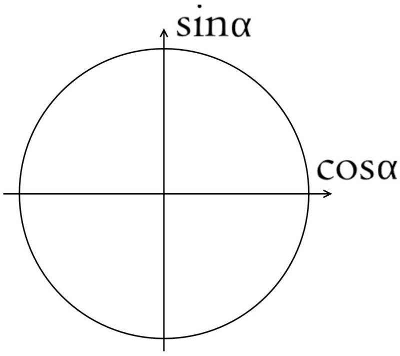


$$\begin{cases} X = X_U - X_S \\ Y = Y_U - Y_S \end{cases}$$

Where:
X, Y is the vector position of the user relative to the screen,
X_S, Y_S is the vector position of the screen relative to the Kinect,
X_U, Y_U is the vector position of the user relative to the Kinect.



The result is a vector that represents the user's position relative to the screen. By normalizing this vector (divide each component of the vector by the norm of the vector $\sqrt{(X^2 + Y^2)}$), the values of the 2 components now directly represent the cosine (x value) and sine (y value) of the angle between the screen and the user. With a simple arccosine (or arcsine) function we can extract the exact value of this angle and verifying the sign of the sine (or cosine) we can finally adjust the rotation to fit between the interval range <0, 360°>.



	$\cos\alpha > 0$	$\cos\alpha < 0$
$\sin\alpha > 0$	$0° < \alpha < 90°$	$90° < \alpha < 180°$
$\sin\alpha < 0$	$270° < \alpha < 360°$	$180° < \alpha < 270°$

Code snippet 4.2

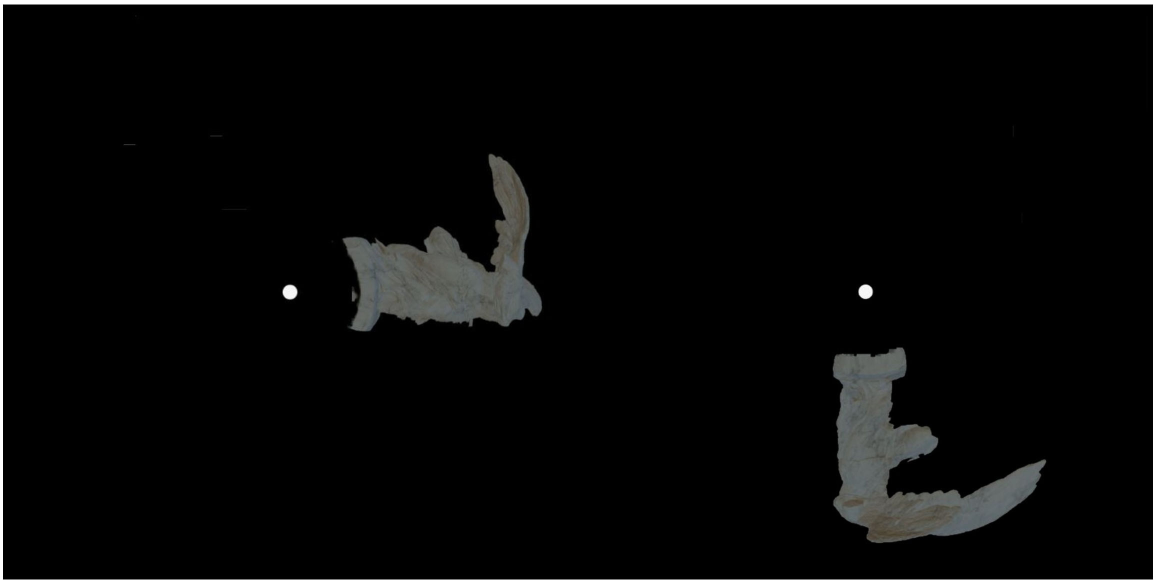
```
// Calculate the direction Vector
Vector2 pointOfViewDirection = UserPosition - ScreenPosition;
// Normalize the direction Vector
pointOfViewDirection = pointOfViewDirection.normalized;

// Calculate the ArcSine of the y value
float angle = Mathf.Asin(pointOfViewDirection.y);
// Convert from Radiants to Degrees
angle *= Mathf.Rad2Deg;

// Correct the angle
switch (pointOfViewDirection)
{
    case var _ when pointOfViewDirection.x > 0 && pointOfViewDirection.y > 0:
        angle += 0;
        break;
    case var _ when pointOfViewDirection.x <= 0 && pointOfViewDirection.y > 0:
        angle = 180 - angle;
        break;
    case var _ when pointOfViewDirection.x <= 0 && pointOfViewDirection.y <= 0:
        angle = 180 - angle;
        break;
    case var _ when pointOfViewDirection.x > 0 && pointOfViewDirection.y <= 0:
        angle = 360 + angle;
        break;
}
```

Extracted, the angle must be applied to 2 different and independent rotations:

- The rotation is firstly applied to the object (effecting the angle around the vertical y axis), to rotate it to show a different side to the camera.



- The image captured by the camera is then rotated around the center of the screen in order to present the reflection to the user.
The two rotations work together to give the user the illusion of continuity of the 3D view, giving at each frame the view of the object from the user’s point of view and moving the image to follow the user’s movements.

CHAPTER 5

Image creation

Once the position of the user is tracked we must create a 2D image of the given point of view of the 3D model we want to display and update this image.
The process of creating a 2D image of a 3D scene is called projection.
The 2D image is then projected onto the cone and to give the user the impression of tridimensionality.
The process of recreating a 3D scene from a 2D scene is called reconstruction.

5.1 From 3D to 2D: image projection

To reach this goal we have a game camera in the game scene that mimics the eyes of the user, the position of the camera will follow the position of the head of the user with the technique we saw in the chapter 4.1, while the rotation assumes that the user is always facing the hologram.
There are few parameters intern of the camera that we need to address, that are already given by the game engine and do not need calibration:
f: focal distance
x, y: focal point coordinates

In the scene the light rays are emitted from the light sources (i.e. pointlights, spotlights but also diffuse ambient light), whose complete lack leads to a totally dark scene, and bounce in the scene and specially onto the 3D object.
The light rays that pass through the camera lenses are captured and their light is "stored" on a 2D plane. One dimension is lost, in fact different points in the scene that belong to the same straight line that connects the center of projection and the point on the 2D plane cannot be distinguished.

The part of the image plane set by the camera parameters is the image that represents the given view (the position and rotation of the camera), with the given parameters of the 3D scene.
However, the image has both the axis flipped because of how the rays are captured, after flipping the image around its axis we obtain the final image that will be showed on the screen and reflected onto the cone.

The equations that link uniquely a 3D point in the scene to a 2D point in the image plane are called "collinearity equations", two equations that link each 2D coordinate to both the internal and external parameters and the 3D coordinates:

$$x - x_o = -f \frac{m_{11}(X - X_O) + m_{12}(Y - Y_O) + m_{13}(Z - Z_O)}{m_{31}(X - X_O) + m_{32}(Y - Y_O) + m_{33}(Z - Z_O)}$$

$$y - y_o = -f \frac{m_{21}(X - X_O) + m_{22}(Y - Y_O) + m_{23}(Z - Z_O)}{m_{31}(X - X_O) + m_{32}(Y - Y_O) + m_{33}(Z - Z_O)}$$

Where the parameters are:
X_O, Y_O, Z_O the 3D coordinates of the point in the scene
x, y the 2D coordinates of the point in the image plane
X, Y, Z the 3D coordinate of the camera in the scene
m the rotation matrix of the camera
f the focal length of the camera
x_O, y_O the 2D coordinates of the main point in the image plane.

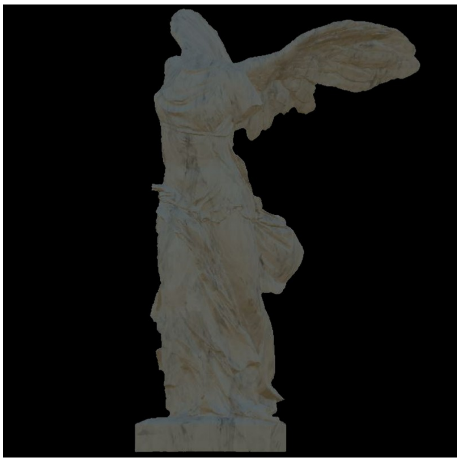
For each point of the object, the relative image coordinate is computed and mapped onto a grid. More than one 3D point can be mapped for the same 2D cell, the point "closer" to the camera is taken (it takes in consideration the occlusions). The final grid is the image that represents the projection of the object onto the plane image with the given camera parameters.

5.2 Image pre-distortion

The obtained image is represents the object from the camera point of view and can be used for example as the scene to be reflected in a “Pepper’s Ghost” or “Hologram pyramid”, as these systems utilize flat glass panels.
But the image cannot be used for our display as it utilizes a cone which does not have flat panels but instead has a curve surface.
The image must be “pre-distorted” before being displayed in order to have a reflected image that is viewed is straighten.
The in-depth process of the pre-distortion is presented in Xuan Luo’s “Pepper’s Cone” that we will now see in general [10].

A series of simple and known colored pattern (i.e. parallel vertical stripes) are displayed sequentially on a screen on which a cone built like the final cone used in the display.
The reflected pattern is captured by a calibration camera placed in the expect position of a user viewing the display.
For each displayed pattern a picture is taken from the camera a.
The distortion map is obtained comparing each pattern with the distorted version recorded from the camera.
This distortion map is then applied to every projection image to extract the distorted version to be displayed.
Because of the inverse process used for the calibration, the reflected image is now perfectly straight on the reflective cone.

As the user moves a new image of the object is projected, distorted and rotated to face the user’s point of view (see chapter 4).



5.2 From 2D to 3D: object reconstruction

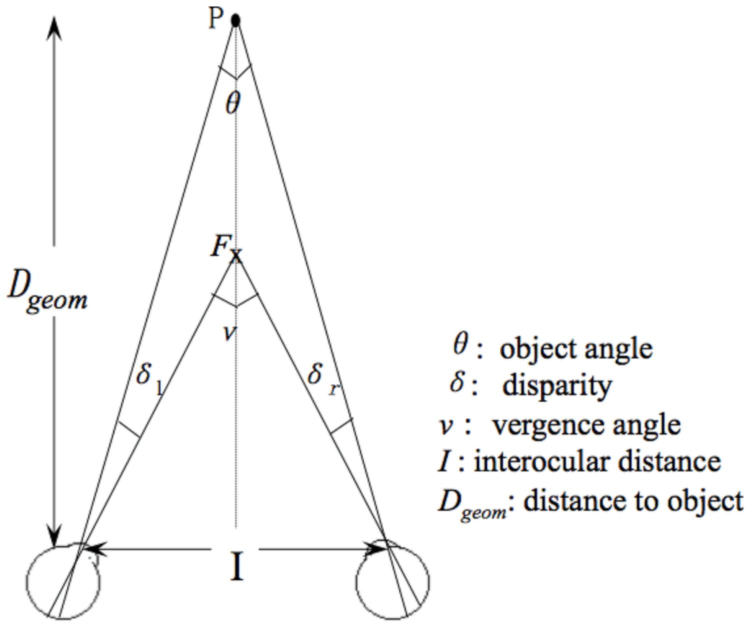
Given the 2D image displayed we must recreate a 3D scene.
Unfortunately, contrary to what we saw with the projection, the reconstruction is not univocal: in fact to each 2D point in the image corresponds a straight line that connects the 2D point and the center of projection, therefore there are infinite points that could potentially represent the 2D point in the 3D scene.

This is also demonstrated by the inversion of the same "collinearity equations" we used before. Inverting the equation we obtain the equations of two planes, whose intersection define a straight line, the same straight line that connects the 2D coordinate and the center of projection.

$$[m_{31}(X - X_C) + m_{32}(Y - Y_C) + m_{33}(Z - Z_C)]*(x_a - x_0) = f[m_{11}(X - X_C) + m_{12}(Y - Y_C) + m_{13}(Z - Z_C)]$$

$$[m_{31}(X - X_C) + m_{32}(Y - Y_C) + m_{33}(Z - Z_C)]*(y_a - y_0) = f[m_{21}(X - X_C) + m_{22}(Y - Y_C) + m_{23}(Z - Z_C)]$$

How can we define uniquely the 3D coordinates?
The human visual system utilizes 2 different cameras (the 2 eyes of a person): the equations of the 2 straight lines intersect at the precise 3D coordinates of the point.
Our brain automatically uses the information given by the two cameras to reconstruct the 3D scene.

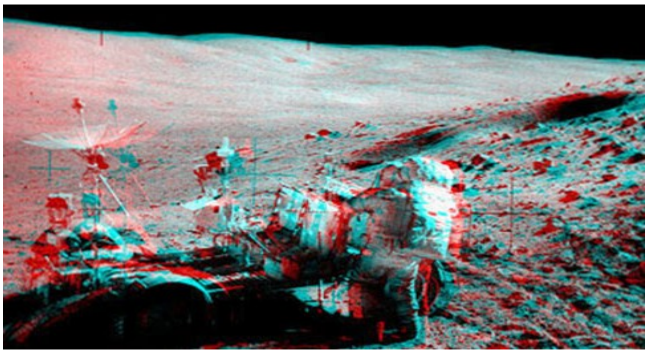


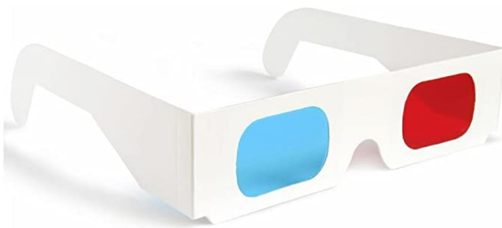
This system is called stereopsis: the perception of depth and tridimensionality obtained through the information given by two eyes (binocular vision).
Generally speaking the brain can find, for each point in the image created by an eye, the corresponding point in the other image.
We can replicate this technique in our game scene by utilizing two separate cameras, the two cameras must:

- be relatively close to each other: closer cameras give less depth perception, farther cameras makes it harder to match the 2D points in the images due to different points of view
- be facing the same way: the optical axis, the straight line that passes through the center of projection and it is perpendicular to the image plane, of the two cameras should intersect in the focus point.

A simplification may assume the two cameras parallel, which is not physically correct but is simplifies a lot the calculus and still maintains a similar effect.
To mimic this system and give two different images from the brain to reconstruct in a 3D scene, we must use two cameras in our scene.
The two images obtained should be presented together, but each to a different eye: it is not possible by simply overlapping the two images, as the eyes will catch both images.

The technique mostly used is the anaglyph: the two images are filtered using chromatically opposite colors (usually red and cyan), encoding each image. The user who looks at the image must wear special glasses with the same colors of the lenses: if the image means for the right eye is colored in red, the lens of the left eye should be cyan: thanks to the tristimulus theory the left eye will not perceive the image and therefore each eye will only capture the image meant for it.





(a) An image example (b) The glasses to view the image

Fig 5.1 Anaglyphs

In our project it would be possible to create such an image and reflect it onto the cone (as demonstrated in the "Pepper's cone" [10]), but it would force the user to wear special glasses, going against our idea of a device that doesn't require any equipment for the user to utilize.
The results follows the simplification of assuming just 1 camera that generate an image for both the eyes.
The difference is hardly noticeable by untrained eye.

CHAPTER 6

Interaction paradigms

The device can even function just as volumetric display, the object is displayed and shown to the user, who can walk and go around the display to view all the sides of the object. An interaction is added to the display in order to interact and make a use of the displayed object.

The Human-Computer Interaction is the study and design of interaction paradigms that allow a computing device and the user to communicate.
To design the HIC of our display we studied the use-case scenario to make a list of the requirements of the interaction pattern.

To better understand the design of the interaction we have to recap the environment and the setup in which the display works and briefly analyze it..
The Kinect sensor used to track he user position is placed in a high position, at least 2 meters from the screen and possibly "hidden" from the user.
The said screen is placed in the middle of a wide empty area in which the user can roam freely.
On top of the screen is placed the reflective cone, and under it the audio output is set (either the Amazon Echo Dot or another speaker).
The computing unit is placed nearby both of the systems as it is connected by wire to the Kinect sensor and the screen (and to the speaker if present).

The display must have a quite unique interaction design as most of the displayed information are not written words but complex 3D objects.
A menu can be displayed but its elements must be of large size as must be seen even at higher distances.
An analysis shows how the usual input systems are not fit for the display.

6.1 Other approaches

Given the setup (see chapter 3) there some restrictions on the possible interaction patterns that can be used:

- Touchscreen (the scree)	the user can't access the screen that projects the image because it is placed underneath the cone, a projected GUI cannot be interacted with with the touch as the cone does not have any touch sensor.
- fixed or wired controller (gamepad or mouse + keyboard)	the user will be constantly moving around the area surrounding the screen: it would hinder the movement with the cable or force the user to go back and forth between the workstation and the display.
- wireless controller (gamepad, mouse + keyboard or smartphones)	the user's movement is not hampered, but a controller would act as a forced intermediate means of communication, a bottleneck similar to special glasses or headgear to view the display: the user interacts with the controller which in turn communicates with the computer and the display. The user doesn't communicate directly with the display.

To make the interaction feeling more "natural", it has been decide to design interactions that mimic the communication between humans: most of the communications that humans exchange are verbal, words are easy to pronounce and to understand to the vast majority of the users.
Gestures are another widely used way of communication: from simple and not essential gestures used to accompany verbal communication, to more complex sets of gestures used i.e. the sign language used to communicate with deaf mute people.

6.2 Voice interaction: Alexa

As we previously saw, Alexa is a virtual assistant present in the Echo Dot.

The Amazon Echo lineup is not the only device that can be connected to the Amazon Alexa service, as the service runs on different devices such as smartwatches, and can be used to create a “smart home environment” to control a household with just the voice: from smart TVs, light bulbs and security cameras to thermostats and speakers to be reachable throughout the house.
The Amazon Echo lineup is equipped with Bluetooth connection, used to connect to the various devices mentioned above, and WiFi, that allows to connect to services such as weather forecast, news, music streaming and even shopping and can be used in any developed program [22].

A program developed (usually in Javascript) for the platform is called a “skill”. To develop such skill, Amazon provided a dedicated service: the Amazon Alexa Skills Kit (ASK CLI).
This service can be fully accessed via browser and contains, among other things, an IDE and a testing section that can simulate an Amazon Alexa and its interaction, in order to cover the entire development process, from the high-level design, to the code writing and testing and finally the distribution.

A skill is particular as devices that run Alexa usually don’t have a screen, and therefore most of the interaction is vocal. A skill must be designed accordingly.

Alexa is designed to respond to a wake word (“Alexa” is the default) and analyze the user’s voice command to react properly.
A skill is programmed to launch at an invocation name, in our case “Pepper’s Virtual Assistant”.
After the skill has been launched particular phrases or commands can trigger intents that are the instructions that Alexa associate to the command.
Alexa is trained to recognize and respond to phrases similar but not identical to the commands provided as samples by the user: this high adaptability proves very useful to the final user, making the conversation more natural and less fixed to a set of phrases [22].

6.2.1 Intents

Each intent is linked to its intent handler, a method that is called whenever the intent is triggered and represents the set of instructions to follow.
For our project we developed the following intents:

Intent	Function	Invocation Sample
StartPresentation	Open the first object in the catalogue to start browsing the catalogue,	“Begin the presentation” -or- “Start from the beginning”
SkipToObject	Start the presentation from a given object in the catalogue,	“Show me the Nike of Samothrace”
GoNext	Open and show the next object or detail of the same object,	“Go to the next object in the catalogue”
GoPrevious	Open and show the previous object or detail of the same object,	“Show me the previous object in the catalogue”
ManageMenu	Show/hide a menu with a list of the objects and the relative details present in the catalogue,	“Open the menu” -or- “What’s in the catalogue?”
ManageDescription	Play/pause an audio description of the currently showed detail of an object	“Tell me something about this” -or- “Play the description”
Assistant	Closes the presentation if open and shows the assistant, the user can then ask for help or close the program	“Assistant!” -or- “I need help”

6.2.2 Communication Unity-Alexa

To make interact the hologram and the invoice controls a connection between Unity Engine and Alexa Development Skill must be established. For this purpose we utilized the library “Alexa SDK for Games” developed by Austin Wilson [23].
This library uses PubNub, a Realtime Communication API, that acts as a bridge between the Alexa Skill and the Unity project.

Connection setup:
This procedure is initiated by the user who launches the Alexa skill. The skill searches for an existing server in the PubNub Cloud services. If either a server is found or a new one is instantiated, the PubNub service returns Alexa a key to connect to the given server and Alexa outputs the key through its screen (Amazon’s testing environment might be needed if the Amazon Echo device does not have a screen).
The same connection key must be inserted in the Unity game scene, allowing it to connect to the same server.
Once the connection is set, Alexa and Unity can exchange messages.
The procedure is summarized in **Fig 6.1**.

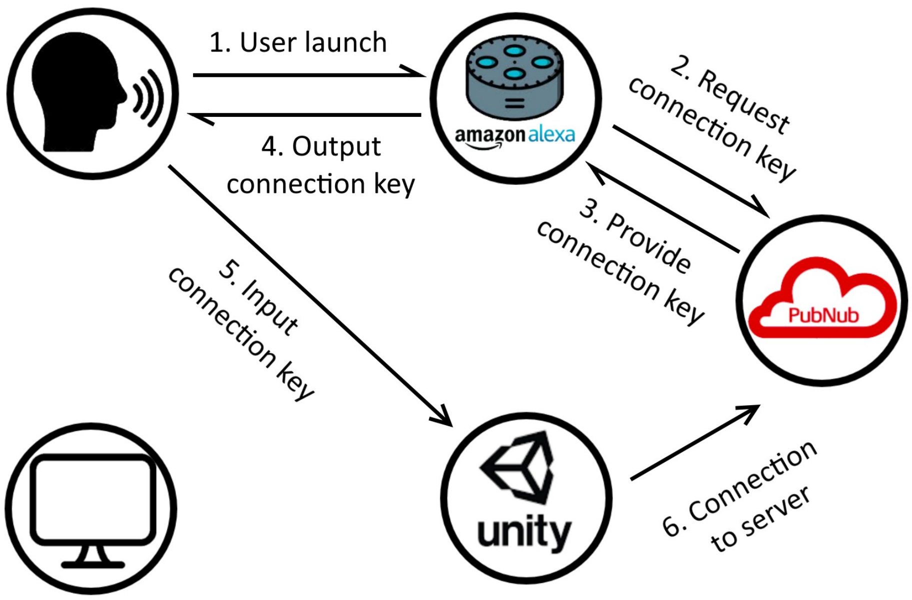


Fig 6.1: setup of the connection

Exchange of messages:
The communication between the Unity game scene and the Alexa skill ALWAYS begins with Alexa and it's triggered by a user's command, which sends a message (a dictionary of <string "type", object "message">) to the PubNub server, and if needed stays listening for the response in the same server.
The Unity scene is already listening in the serve and whenever a new message is published an event is called: it then extracts the type and the message from the received event and acts accordingly.
If needed it then prepares a message with the same structure of the received and publishes it on the PubNub server, for the Alexa Skill to receive.
The procedure is summarized in **Fig 6.2**.

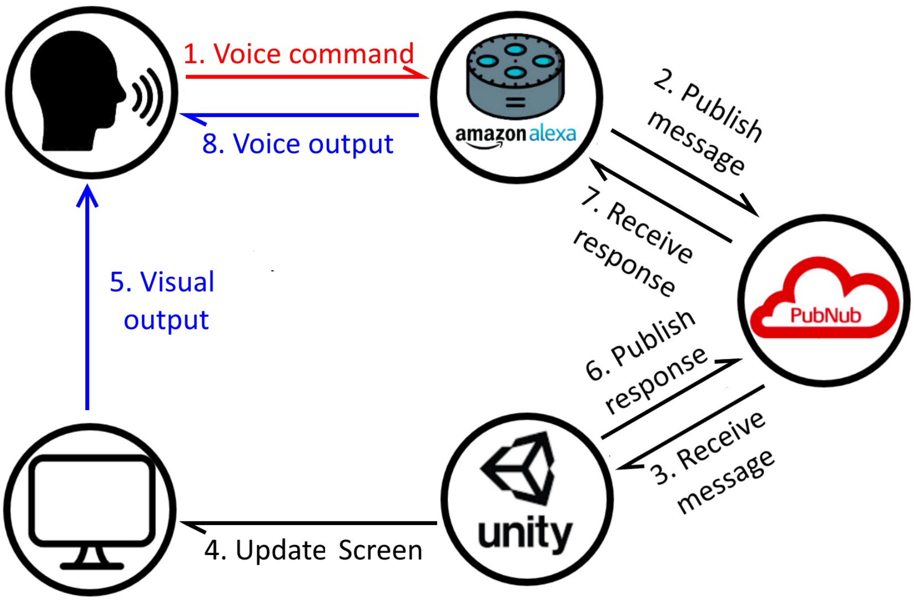


Fig 6.2: messages exchange and output

Code Snippet 6.1
Skip to Object:
In this example the user asks Alexa to skip to a given object.
Alexa prepares the message (<"Skip", "Name of the object">) and publishes it on the server, then stays listening for the response.
Unity recognizes the message type "Skip", and searches for the object contained in the message. It finds it and prepares the message to send back (<"response", "true">) and publish it. Alexa receives the response and its message that states that the object has been found. She tells the user the object is in fact in the catalogue.

Code snippet 6.1

Alexa side

```
const SkipToObjectIntentHandler = {
    canHandle(handlerInput) {
        return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
            && Alexa.getIntentName(handlerInput.requestEnvelope) === 'SkipToObjectIntent';
    },
    handle(handlerInput) {

        const {requestEnvelope, responseBuilder} = handlerInput;
        const ObjectName = Alexa.getSlotValue(requestEnvelope, 'ObjectToOpen')

        var payloadObj = {
            type: "Skip",
            message: ObjectName
        };

        await alexaPlusUnity.publishMessageAndListenToResponse(payloadObj, attributes.PUBNUB_CHANNEL, 5000).then((data) => {
            switch(data.message.response)
            {
                case "true":
                    return response = handlerInput.responseBuilder
                        .speak(handlerInput.t("Tho object has been found in the catalogue"))
                        .getResponse();
            }
        }).catch((err) => {
            // Handle errors
        });
    }
};
```

Unity side

```
public void OnAlexaMessage(HandleMessageEventData eventData)
{
    Dictionary<string, object> message = eventData.Message;

    switch (message["type"] as string)
    {
        case "Skip":
            string objectToOpen = message["message"] as string;

            // Checks if the "objectToOpen" is in the catalogue, and finds it

            string responseText = "true",
                   responseType = "response";

            Dictionary<string, string> messageToAlexa = new Dictionary<string, string>();
            messageToAlexa.Add(responseType, responseText);

            AmazonAlexaManager.SendToAlexaSkill(messageToAlexa, OnMessageSent);

            break;
    }
}
```

6.3 Gestures recognition: Kinect

Kinect is used to track the head of the user to follow his position and rotate accordingly the hologram.
It is also utilized to track the hands of the user, which he can use to interact with the hologram itself.

The gesture interactions depends on the state of the device:
- Menu State: the user is in a menu, like the language selection menu or the catalogue menu.
An icon appears on the display that follows the position of the hand of the user, mimicking the mouse movement. Hover the icon over a button to select it and hold the hand in position for few seconds to click and press the button.

- Browsing State: the user has started the presentation and a 3D object is displayed for the user to see.
5 invisible triggers are placed between the user and the display (since the Kinect cannot recognize the front from the back of the user, we assumed the user always facing the display: an assumption that can be accepted given the setup and the purpose of the project), various hand movement collide with the triggers and prompt different commands

To better understand the browsing hand gestures, visualize the catalogue as the table in **Fig 6.3**: per rows there are the objects and per columns there are the details.
The hand swipe movement is used to move to a nearby cell.

	Detail 1	Detail 2	…
Object 1: Nike of Samothrace	Reconstruction of the head	Reconstruction of the left arm	…
Object 2: Venus de Milo	Reconstruction of the arms	Reconstruction of the clothes	…
…			

Fig 6.3: the catalogue

6.3.1 Recognized gestures

Each gesture is recognized based on the collision between the 5 triggers and the tracked hands of the user:

For our project we developed the following gestures:

Name	Function	Movement
Hold Center	Play description: an audio description of the currently showed detail of an object is played and output through the speaker. Pause description: the audio description that is currently playing is stopped. To play the description again the user must first release the position and then stretch the hand forwards again. If the description is played again, it starts from the beginning.	The user stretches the hand forward and the hand collides with the central trigger. The user holds the position for a few seconds.
SwipeLeft	Open previous detail: opens and shows the previous detail of the same object. If the first detail is currently showing, it will open the last detail of the same object	The user stretches the hand forward and the hand collides with the central trigger. Then the user moves the hand to the left to exit the central trigger and collide with the left trigger.
SwipeRight	Open next detail: opens and shows the following detail of the same object. If the last detail is currently showing, it will open the first detail of the same object	The user stretches the hand forward and the hand collides with the central trigger. Then the user moves the hand to the right to exit the central trigger and collide with the right trigger.

Swipe Up	Open previous object: opens and shows the previous object. If the first detail is currently showing, open catalogue menu.	The user stretches the hand forward and the hand collides with the central trigger. Then the user moves the hand to upwards to exit the central trigger and collide with the upper trigger
Swipe Down	Open previous object: opens and shows the previous object. If the last detail is currently showing, open catalogue menu.	The user stretches the hand forward and the hand collides with the central trigger. Then the user moves the hand to downwards to exit the central trigger and collide with the lower trigger.

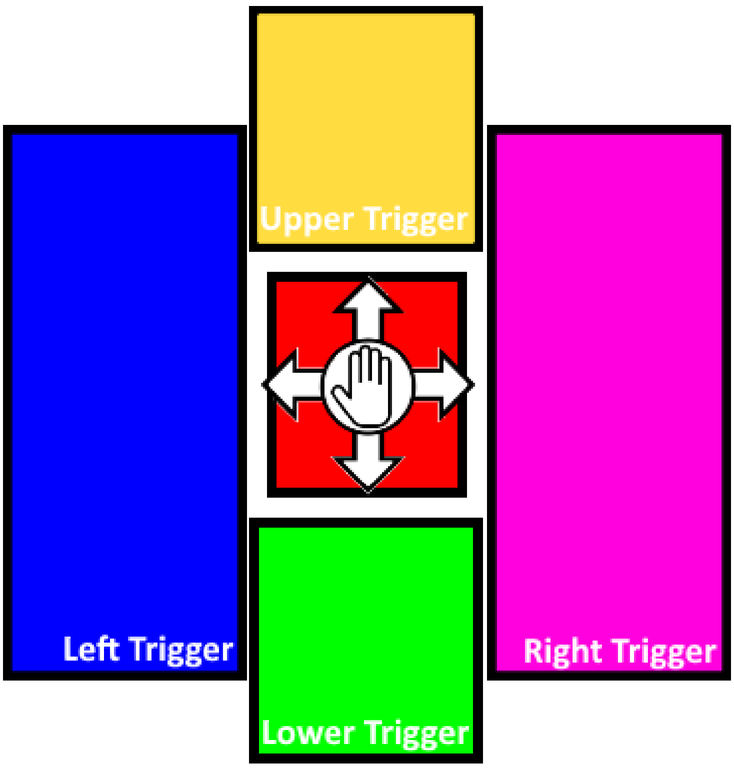


Fig 6.4: the triggers position

6.3.2 Text to speech synthesis

Even tho Alexa provides a robust text to speech synthesizer, it has been decided to keep the two approaches separated and to have them independent from one another.
We opted to use the “IBM Watson SDK” [24] as it provided the most natural and pleasant voice output.
The service connects to the IBM Cloud, and allows to highly personalize the output voice, from its pitch and tone, to its accent.

The service contains a set of preset voices aimed at covering most use case and in many different languages, like “en-US_AllisonVoice” and “it-IT_FrancescaVoice”.
It is else possible to create a new voice and customize it by various parameters:
- gender: the gender of the voice, <male> or <female>,
- language: the language and region of the voice, i.e. en-US,
- pronunciation: the pronunciation of language, can be used to create accents,
- volume: the speak volume, determines whether the voice is loud or quiet
- pitch: the highness or lowness of the voice tone, determines whether the voice is deep or high-pitched,
- speed: the speed with which the text is read.

[24]

We tried to create a pleasant voice, easy to understand and a various tone range to give a natural feel and avoid plane voice synthesis effects.

6.4 Language localization

The device provides a system of localization to come meet the needs of users of different countries. In fact the names of the objects, the vocal commands, and the descriptions are all translated in both Italian and English, and any other language can be added: Amazon Alexa and IBM Watson provide their own translation system, while the strings are manually translated to reduce the errors of automatic translations [12][25].

To select the language:
- Alexa interaction: the language is automatically selected from the settings of Amazon Echo Dot. To change the Amazon Echo Dot Language open the app from a smartphone, then open Settings > Device Settings, select the device ("Amazon Echo Dot" is the default name) > Language > select the language.

- gesture interaction: move your right hand to move the cursor. Hover the cursor over the selected language and keep the position for 3 seconds.

As the setup happens at the launch of the app, both the interaction paradigm and the language cannot be changed during the session.
To change either of them, you must relaunch the program.

CHAPTER 7

Applications

The uses that can be made of the device we presented various, as the device allows to display a 3D object and view it from all its angles, so it could prove useful in situation where it would be needed.

In our project we implemented two possible applications that well suited the capabilities of the volumetric display.

7.1 Virtual gallery

This is the primary use we had planned for the display, as a virtual gallery for a museum exhibition.
The display allows to view a 3D object from various points of view so it comes natural to use this display to show objects for the users to see.
While it is obviously better to admire a sculpture in person, this display enables users to view different objects without the need to move between them.

In our example we used a simple model of the Nike of Samothrace, a Hellenistic sculpture that has been damaged over time, and recreated the missing parts with the software Blender, allowing the user to explore the sculpture from various perspectives, with particular attention to the reconstructed details.
DISCLAIMER: the reconstruction does NOT match the real reconstruction made by historians, but was the figment of the imagination of the author. Pieces of different statues were put together to recreate a fake sculpture. Therefore the descriptions are fictional too.

Image	Description
	[ENG] - The Nike of Samothrace is a 245 cm high marble sculpture of the Rhodian school, attributed to Pythocritus, datable to 200-180 BC. approximately and today preserved at the Luvre Museum in Paris. [ITA] - La Nike di Samotràcia è una scultura in marmo alta 245 centimetri di scuola ròdia, attribuita a Pitocrito, databile al 200-180 a.C. circa e oggi conservata al Museo del Luvre di Parigi.
	[ENG] – This is a fictional reconstruction made to show the capabilities and possibilities of our device. [ITA] – Questa è una ricostruzione fittizia fatta per mostrare le capacità e le possibilità del nostro dispositivo.

	[ENG] - The Winged Victory wears her hair gathered as appropriate for the time. The serious gaze is fixed on the enemies, ready for war and triumph. [ITA] - La Vittoria alata porta i capelli raccolti come consono al tempo. Lo sguardo serio si fissa sui nemici, pronta alla guerra e al trionfo.
	[ENG] - In her left hand he carries a sword. In fact, Nike represented the personification of Victory, both sporting and military. [ITA] - Nella mano sinistra porta una spada. La Nike rappresentava infatti la personificazione della Vittoria, sia sportiva che bellica.
	[ENG] - The right hand instead encourages the soldiers to follow her towards the battlefield and the honor that comes from Victory. [ITA] - La mano destra incita invece i soldati a seguirla verso il campo di battaglia e l'onore che deriva dalla Vittoria.

7.2 Virtual assistant

Amazon Alexa is a virtual assistant that consist only in a voice to interact with.
An idea for the device was to give Alexa a face and a body, for the user to talk to and interact with.

We used a 3D model and animated it with Adobe's Mixamo[1], to create a hologram that could respond to our requests giving a more natural feeling.
The virtual assistant can help the user to navigate the gallery, can provide help and demonstrate how to use the gestures to interact with the hologram itself.

A similar use case could be a virtual museum guide: the display, placed at the center of the hall or a room of the museum can interact with the visitors, guiding them around the exhibition, explaining details of the objects around while pointing directly at them.
An example could be a virtual Leonardo da Vinci that acts as the guide in an exhibition of Leonardo da Vinci's works.

Fig 7.1 The virtual assistant

[1]https://www.mixamo.com/

CHAPTER 8

Conclusion

In terms of cost, the device combined ranks in the average of the many devices already present on the market and that we discussed, though it requires an additional computing center that the other solutions have incorporated or don't need at all.
But compared to the other devices, our was the solely capable of presenting to the user a full 360° view of the displayed object without any hidden angle of view.
Even tho the Pepper's Display has no issues with the horizontal position of the user, the verticality is a problem as the user's eyes must be at the same height of the center of the projected image on the cone.

Between the various techniques we considered to track the user's position, the Kinect sensor is undoubtedly the best as it checks all the requirements for a smooth tracking, while its problems can be worked around with a studied set-up environment.

The interaction seemed fluid and organic: the conversation with a Alexa with a visual and animated body added naturalness to the experience.

8.1 Future works

The display can be further expanded to add features that currently it does not provide.

8.1.1 Multiuser

The display we developed has one main issue: it can be used by only one person at the time, while being viewable by multiple users would be preferable.

A solution could be creating multiple images, one for each user, and make each image follow its user. This solution would have many problems:
- overlapping images: having many images could lead to the overlap of some of them, i.e. two people close to each other would create two images that would collide.
- safety of the users: while the user explores the 3D object his focus is towards the hologram, while paying less attention to the surrounding environment which comprehends other users who are moving too.

A solution that solves both these two problems would be to stop the hologram in case two users get too close: for safety reasons there is no more possibility of images colliding.

Keep in mind: our display utilizes the Kinect device, which has a maximum number of concurrent users (see chapter 4.3).

Figure 8.1 Multiple images for multiple users

8.1.2 Augmented object

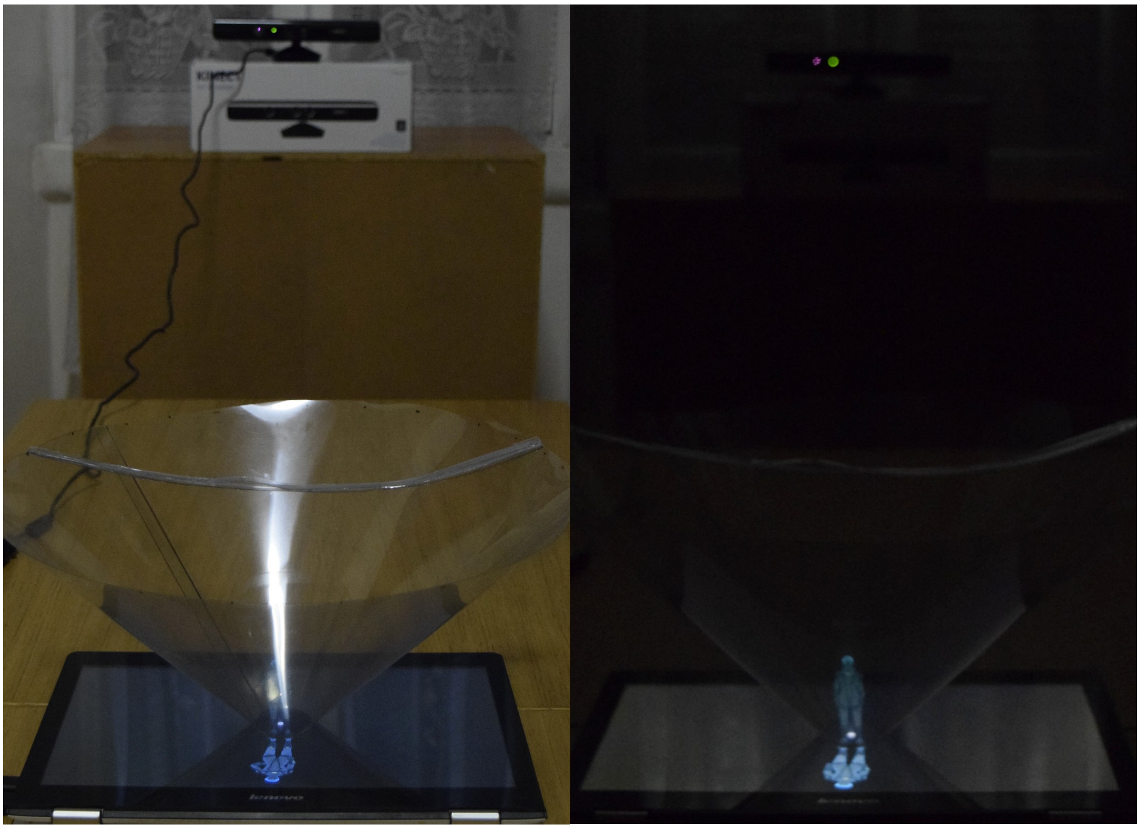
The display consists of an empty cone place on top of a screen.
This empty space can be filled with an object, i.e. a sculpture like the Nike of Samothrace, and use the hologram to augment the object, i.e. projecting only the missing parts onto the real sculpture.
This technique could prove useful in presence of an object of high value which would be worth to see by itself. Adding the augmented objects would add value to the view of the original object itself.

This Augmented object setup would have to face different problems:
- the hologram require low levels of light to be fully visible, but with low lights the object to augment would be hard to see, the lights should be placed and regulated so that both the hologram and the object are visible, while not being reflected on the cone, disturbing the user’s view.
- static environment: placing an object would mean being able to show just virtual objects related to the augmented object.
With the right lights, it could be possible to show the augmented object only when needed.
- scanning the augmented image: to correctly show the virtual objects, it is necessary to hide the parts of these objects that are occluded by the augmented object: this requires to scan the augmented object and apply to the virtualized object a special shader that makes it not visible.
There are many techniques and devices that can do the process of 3D scanning.

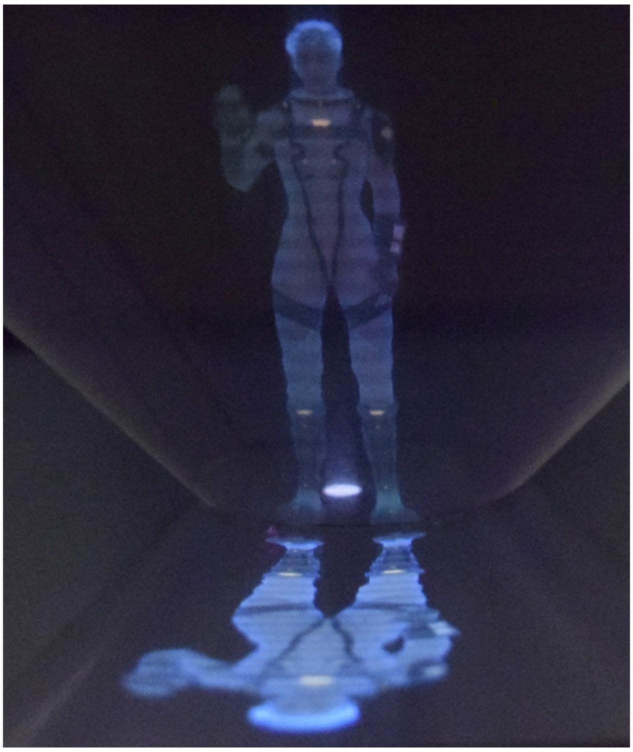
Figure 8.2 Augmented Nike of Samothrace

CHAPTER 9

Images



The setup with lights on and off



The virtual assistant

The projected image side-by-side the resulted reflection

BIBLIOGRAPHY

[1] Wikipedia. *Holography*, as 8th September 2020.

[2] Holocenter. *What is holography?*, 2020.
http://holocenter.org/what-is-holography

[3] D. H. Close, A. D. Jacobson, J. D. Margerum, R. G. Brault, and F. J. McClung. *Hologram recording on photopolymer materials*, 1969.
https://doi.org/10.1063/1.1652756

[4] Adam James Schwarz, *Considerations on volumetric 3D display systems*, 1994. oai:ir.canterbury.ac.nz:10092/7572

[5] Saoirse Kerrigan, *13 Hologram Projections Around the World That Are More Than Just a Pretty Display*, 2018.

[6] Wikipedia. *Volumetric display*, as 13th September 2020.

[7] Daniel Smalley, Ting-Chung Poon, Hongyue Gao, Joshua Kvavle and Kamran Qaderi, *Volumetric Displays: Turning 3-D Inside-Out*, 2018.

[8] Oliver Eugene Simon and Roger Andrew Santos, *Volumetric Display Research*, 2018. oai:digitalcommons.wpi.edu:mqp-all-3386

[9] Looking Glass, *How the looking glass display works*.
https://lookingglassfactory.com/tech

[10] X. Luo, J. Lawrence, S.M. Seitz, *Pepper's Cone: an inexpensive Do-It-Yourself*, UIST 2017.

[11] Wikipedia, *Kinect*, as 31st August 2020.

[12] Wikipedia, *Amazon Alexa and Amazon Echo*, as 11th february 2020.

[13] Sergio Mascetti, *Corso di Sviluppo di applicazioni per dispositivi* mobili, 2020.

[14] Olivier Magneau, Patrick Bourdot and Rachid Gherbi, *Positioning and identification of markers for 3D tracking*, 2004. https://doi.org/10.1051/meca:2004024

[15] Raffaella Lanzotti, *Corso di Visione artificiale*, 2020.

[16] Jeremy Jacquot, *How the Wii Balance Board Works*, 2010. https://electronics.howstuffworks.com/wii-balance-board.html

[17] Julien Audiffren and Emile Contal, *Preprocessing the Nintendo Wii Board Signal to Derive More Accurate Descriptors of Statokinesigrams*, 2016. doi: 10.3390/s16081208

[18] Kipp Jones, *Using Wi-Fi for indoor positioning*, 2020. https://www.skyhook.com/blog/using-wifi-for-indoor-positioning

[19] Wikipedia, *Wi-Fi positioning system*, as 31st august 2020.

[20] Z. Zhang, *Microsoft Kinect Sensor and Its Effect*, in IEEE MultiMedia, vol. 19, no. 2, pp. 4-10, doi: 10.1109/MMUL.2012.24. 2012

[21] Roanna Lun, Wenbing Zhao, *A Survey of Applications and Human Motion Recognition with Microsoft Kinect*, 2015. doi:10.1142/S0218001415550083

[22] I. Lopatovska, K. Rink, I. Knight, et al. *Talk to me: Exploring user interactions with the Amazon Alexa*. Journal of Librarianship and Information Science, 2019. doi:10.1177/0961000618759414

[23] Austin Wilson, *Alexa plus Unity*, 2018.
https://alexaplusunity.readthedocs.io/en/stable/index.html

[24] IBM Watson, *Text to Speech*, 2020
https://cloud.ibm.com/apidocs/text-to-speech

[25] Graham Templeton, *Language Support in voice assistants compared*, 2020.

IMAGES REFERENCE

[Fig 2.1] Holograms on a banknote
SUSANNE HESPING, Prien Marketing GmbH

[Fig 2.2] From 3D to 2D: how to create a hologram
Chris Woodford. https://www.explainthatstuff.com/holograms.html

[Fig 2.3] Pepper's ghost effect:
(a) An explanation of the effect,
Eric Pierce, Pepper's ghost low angle, 9th March 2005.
https://commons.wikimedia.org/wiki/File:Peppers_ghost_low_angle.jpg

(b) A famous use of the technique
Digital Domain, 2012. https://theundefeated.com/features/the-strange-legacy-of-tupacs-hologram-after-coachella/

[Fig 2.4] An example of a holographic fan
Garsent, Proiettore Ologramma 3D, 2020. https://www.amazon.it/Proiettore-Ologramma-Holographic-Supporto-100- 240v/dp/B07QDYC722

[Fig 2.5] Two examples of pyramid holograms:
(a) DIY pyramid with a smartphone,
mSCI, How to Make a holographic pyramid, 2016.
https://www.youtube.com/watch?v=ucbuc7VXfBI
(b) A full system pyramid
Dreamoc 3L3, Interactive Studio, 2019

[Fig 2.6] Two different volumetric displays:
(a) A butterfly made of floating points of light
DigInfo.tv, Aerial Burton True 3D display, 2014
(b) The Looking glass
Tested.com, Hands-On: Looking Glass holographic display, Looking Glass factory, 2020. https://lookingglassfactory.com/

[Fig 4.1] the tracking the movement of people.
CRCV. Ufc center for research in computer vision, 2019

[Fig 5.1] Anaglyphs:
(a) An image example,
John Watts Young working on Rover during Apollo 16, US, 2011.
https://www.usgs.gov/media/images/apollo-16-john-watts-young-working-rover