Politecnico di Torino

Master of Science

Mechatronic Engineering

Design and implentation of a mixed-reality robotic game

Supervisor:

Prof. Fabrizio Lamberti

Candidate:

Piero Baldo

October 2019

Torino

Contents

1 Introduction

The work carried out in this thesis consists in the implementation of a mixed reality robotic game. This thesis therefore has two objectives: the first is the implementation of a robotic game that follows the PIRGs philosophy (physical intective robo-games), whereas the second is to investigate how players' perception is influenced by the use of tangible objects. With regard to the PIRGs philosophy, it is a guide to how to develop a robotic game. For years, technology has allowed us to sell increasingly sophisticated robot toys to the public, however it has been noticed that the public quickly lost interest even for the most sophisticated robot: Indeed this technology was often expensive, decontextualized and left to itself; ultimately, after the functions that were already pre-programmed to the robot, it was no longer able to provide stimuli and keep up the customer's interest as it does in today's traditional video games. With the application of the PIRGs model it was discovered that robots, even out of production, took on a new life. In fact the PIRGs model with its guidelines aims to give a context to the robot that can better enhance its capabilities through the use of external tools such as, e.g. augmented reality, external tracking systems that allow to interact with the robot in the most natural way possible or the use of physical objects that can interact with the virtual world. This thesis deals with an implementation of a robotic game that follows the PIRGs model; in particular a Cozmo robot, a Kinect V2 and a projector were used.

Cozmo small size (Cozmo's size is a palm of hand) suggests to implement a limited playing area, this limits the movements that players can do, for this

as user interface was decided to use Cozmo cubes, which can be moved by players like in a board game. The great expressiveness of Cozmo makes its an ideal candidate as a robot in a PIRGs game. In fact its ability to imitate human expressions such as sadness, joy, anger etc. help to create a bond with the player who will be led to consider the robot not a simple machine but a real playmate, (this feature is one of the central points of the PIRGs guideline).

The Kinect was used instead to locate the cubes on the game board by means of its RGB camera, it was placed above the playing surface, so that from position it can trace the cubes correctly. The localization of the cubes has been implemented with the use of the OpenCV library using an algorithm able to detect the colours; for this reason visual markers have been applied to the cubes, so that the algorithm can easily extract their colour from the background and locate the cubes.

The projector was instead used to project the virtual gaming area; players interact with both the robot and the virtual environment through the user interface (cubes tracked by Kinect). To create the game environment Unity game editor was used, instead the robot logic and the tracking algorithm were implemented using the Python programming language. The two processes ()Python module and Unity editor) were put into communication using a socket, implemented with the ZeroMQ open-source library.

The game was designed by following a competitive multi-player scheme, where the role of the robot is to arbitrate between the two players. At the beginning of each round, Cozmo communicates, using its integrated loud-

speaker, a series of colours chosen between red green and blue; the players must therefore hit a virtual ball by setting, with each bounce of the ball on their own cube or that of the opponent, the right colour according to the order of the series announced by Cozmo.

Example:

Cozmo says red green and blue; the first player to receive the ball sets his red cube, the opponent responds instead using the green colour; finally the first player ends in blue. If the players manage to finish the series without losing colour, then the current round ends and Cozmo will add new colours to the series. The players, during the round, can ask Cozmo for suggestions; in this case Cozmo will suggest the exact colour of the sequence to the player. To investigate how the payers' perception regarding game system is influenced by the use of tangible objects, three different versions of the game have been implemented, each of which becomes the gesture that players must perform to change colour of their cube and ask for the suggestion.

A set of experiments were carried out by involving several volunteers, offering players a questionnaire at the end of their gaming experience. The questionnaire evaluated the system usability of the game, user's experience with tangible objects and their perception of the robot in the game experience. In summary, based on results it emerged that the three versions differ greatly in regard to SUS (System Usability Scale), the perception of the game changes slightly with respect to the ease with which players can change colour to their cube or ask for suggestions from the robot; finally, as far as the perception of the robot is concerned, it seems from the results that

the use of different gestures by means of tangible objects does not change the perception that the players have of the robot.

2 State of art

In this chapter a brief summary of the history of toy robots is presented, going from the first generations to the robots currently on the market, then analysing the characteristics and specifications that characterize the modern toy robots that producers have come to develop and understand over the years and developments in the sector.

We will then briefly analyse the evolution of the concept of robotic game up to the most modern architectures, presenting different works carried out by students in different universities to which this work was inspired.

2.1 Brief history of robotics

'Robot' is a word coined by Czech playwright Karel Capek to indicate unintended work. The word was introduced in his theatrical work "R.U.R. (Rossum's Universal Robots) "staged in January 1921. However, Capek's merit is limited to being the inventor of the word robot, since the machines he talks about in his work are built according to alchemical / chemical processes Indeed, we can define the robot in modern language as a reprogrammable electromechanical system, with perception capacity and its own intelligence, designed to perform a large number of different tasks. It is not surprising that we can already find the word robot in the early 1900s, in fact the word automaton and android today indicated to define robots with a shape similar to the human one ,they derive from the Greek language, the first from automatos, "which acts of its own will" the second from aner, andros which

means man, here understood as in human form. In fact, it is told in a myth that you belong to Greek culture, that Hephaestus created three automata to help him in his daily work.

The idea of creating autonomous and intelligent mechanical systems is quite ancient and represents the synthesis between the dream of imitating Nature and the need to build machines useful for life and work.

We can find the first traces of applied robotics with scientific rigour in the Hellenistic age of ancient Greece However, the first experiments of self-propelled machines were intended to amaze and impress (in fact they were mostly used for theatrical goals) instead of having really useful purposes in everyday life. Furthermore, environmental limits were imposed because the position of the objects with which the robots had to interact was known a priori, only later with the introduction of artificial vision and machine learning, the robots acquired an autonomy that reduced environmental constraints . The interaction between humans and robots begins at the beginning of the 80s with the introduction of so-called service robotics, service robots are no longer confined to specific areas but interact directly with human beings. Service robots are used in environments that are not easily accessible to human beings, such as the space or submarine environment or capable of carrying out dangerous work such as bomb robots. Robots used in domestic assistance are also being studied, such as cleaning dishes served at the tables or cooking lunch and dinner. Among the service robots that most concern this work are certainly the social robots that have the purpose of entertaining, of which the toy robots are part and which are gaining more and more success

and interest on the part of of the public.

Modern robots can be divided into three macro categories:

- Industrial robots (Figure 1 for an example), they are the first in order of application, today have a vast use in all production sectors. They mainly consist of mechanical manipulators capable of performing certain tasks in the field of industrial production, they are employed in assembly control, movement inspection, etc.

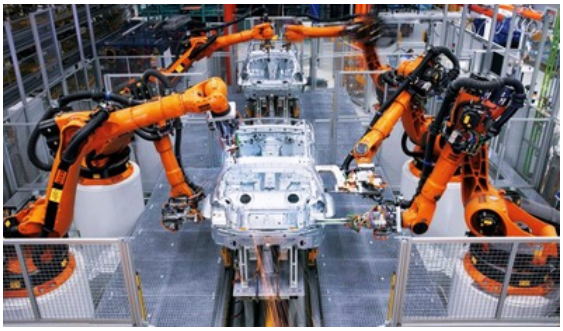

Figure 1: Industrial robot

- In the medical field, the use of robotics is revolutionizing both the surgical practice (Computer Aided Surgery) and that of the clinic (physiotherapy, technological assistance). In recent years robots are being developed that can help doctors during surgical operations using devices non-invasive. The Da Vinci system shown in Figure 2 is noteworthy as it allows the surgeon to operate through a dedicated workstation as if his hands were the pincers of a complex robotic system.

Figure 2: Example of Da Vinci system

- Social robots (see Figure 3) identify a new application of robotics destined to be a tool for social interaction in the future. Once the safety barrier has been eliminated, robots can become part of social life through an accurate set of standards and certifications. They are mostly in the experimental phase, the tasks of these robots range in different areas, from entertainment to assistance for the elderly or sick.

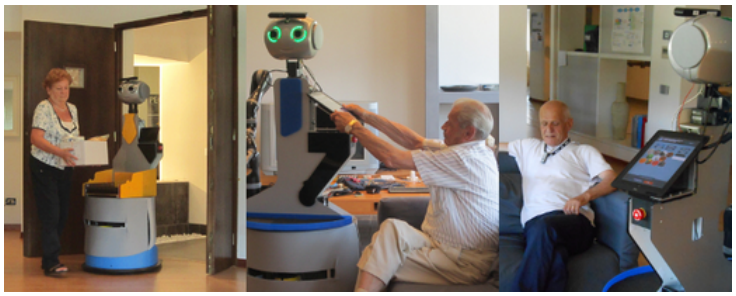

Figure 3: Example of social robot

2.2 First generation of toy robots

In this section, three different commercial robots have been chosen which, according to the candidate's opinion, represent significantly the first real

attempt to insert toy robots in the market, products intended for sale to the general public. These robots, although still rough in their trait,they have the features we can find in the most modern products. Robots must therefore intelligently combine different technologies to interact with the real world in the most natural way possible, especially if they are toy robots because they have to maintain the illusion of being somehow alive and are intended for an audience that may not be experts in the field. In the history of toy robots, only now technologies have come to the reserve point of selling robots to the general public at guaranteed low costs, in this chapter we will focus on three robots in particular because they are the first to have features that may be interesting for the work done in this thesis. Lego MindStorms robot construction sets and Furby interactive robotic pet by Tiger Toys, and Sony Aibo robot dog [**?**].

2.2.1 Furby

Figure 4: Furby robot

Furby is a robot released in 1998 by electronic robotic (see figure 4), is a stand alone, interactive robot, has several sensors and actuators:

- light sensor;
- IR sensor;
- IR transmitter;
- microphone;
- speaker;
- belly touch sensor;
- tongue touch sensor;

Through the use of engines it can move ears, mouth and eyes; Furby is part of the category of pre-programmed games, ie the user is not expected to implement new functions. Furby has the shape of a fancy animal, a kind of cross between a hamster and an owl, obviously designed for a public of children, it obtained a huge success with more than 12 million copies sold. It therefore represents one of the first experiments of social robots with a playful aim widely sold on the market, in fact it is among the first products able to surprise customers because it is able to surprise them with unexpected behaviors and reacting to certain external or environmental stimuli or by the user himself. Surprisingly Sherry Turkle has found that children categorize their Furbys in a new way: "Children describe these new toys as they are emotionally connected to the Furby might be emotionally attached to them

” (Turkle 2000) [**?**]. This consideration puts the toy robots in an unexpected context, in fact they are half way between the animated and the inanimate, suggesting that future developments of these types of products could pose complex problems regarding ethics.

2.2.2 Lego Mindstorms

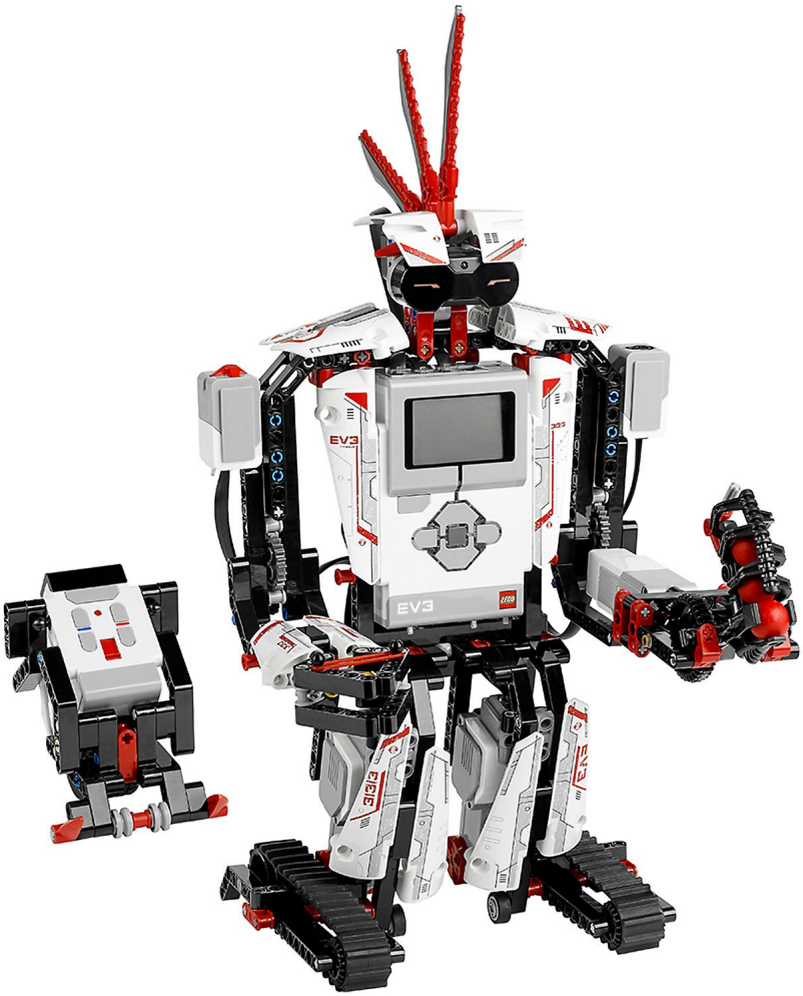

Figure 5: Lego Mindstorms robot

Lego Mindstorms, shown in Figure 5, is a line of Lego products with programmable bricks to allow the user to exploit the potential of the lego product by combining it with the automation provided by the electronic components such as sensors and other interactive gears. It can be programmed using a computer to create virtually all existing electromechanical devices such as elevators or mechanical arms. The first generation of Lego Mindstorms was built around a programmable morning called RCX which can be programmed using different programming languages, including some created ad hoc by Lego like RCX code or ROBOLAB based on the Labview language, otherwise it supports more common languages like C , C++ or Java. Lego Mindstorms, unlike Furby, was born as a programmable robot, that is, the customer had the incentive to write his own program or assemble his own robot with the limitation only of his imagination and the pieces supplied by the product. Despite the rather high cost it has had an immediate success both among adults and children, but not only, lego Mindstorm is in fact also used in schools and universities in laboratories and as an aid to teaching.

2.2.3 Aibo

Figure 6: Sony's Aibo robotic dog

The third example, Sony's Aibo robotic dog in Figure 6. Aibo is one of the many robotic animals developed by Sony, with this name several models have been developed since 1999. Aibo is the evolution of the Furby robot, in fact the Aibo was used by the Sony as the basis for artificial intelligence programs, in fact it represents a platform comprising camera sensors and a functioning mechanical movement system. Indeed, Aibo can react to vocal commands by recognizing the external environment through the SIFT machine vision algorithm, and what is more interesting is the Aiboware algorithm which is able to make the robot grow starting from the puppy phase up to the adult phase by modifying the personality based on the interaction with the owner. Aibo was developed in Japan, the culture of Japan is in fact characteristic

for its respect for the robots, so Aibo deepens the ethical problems already encountered with Furby, since unlike Furby, who is not a real animal and is still too raw in its functionalities to consider it alive, Aibo instead imitates in everything and for everything a living creature like the dog, which has already for some time a strong interaction with the human being.

2.3 Lego Mindtorms and Furby: A comparison

These two different types of toy robots were both introduced in 1998, and had a great impact though in two completely different ways. Lego Mindstorms presents itself as a programmable robot with aspects geared towards education, in fact it is a product that combines well with a philosophy of continuous expansion and support from users, in fact it is possible to purchase different versions to combine the pieces together. On the contrary, Furby presents itself as a stand alone product with its functions already implemented and not expandable, its purpose is that of entertainment even if perhaps involuntarily Furby much more than Lego Mindstorms underlined the ethical problems that can be found introducing robots in everyday life and in children's recreational activities. Another contrast regards the transparency and the opening of the software, initially Lego Mindstorms was born as a closed system, the users could program their pieces but always following the rules of the manufacturer, only later the Lego authorized the users access to the 'RCX's rom, allowing you to create a community and a platform that supports a large number of languages and operating systems. On the contrary, Furby has remained a closed system, its functions have never been fully

documented, in fact not allowing the community to reprogram the robot.

2.4 Ethics

It should come as no surprise that with the introduction of toy robots we began to talk seriously about Ethics. In fact These robots are capable of unexpected actions by the public, as it was written before the children considered Furby not only as a robot, but as something more, in fact an emotional relationship was established between the child and the toy, they must also be sure as they interact continuously with the real world but physical security alone is not enough, in fact in recent years the "Roboetic" neologism has been coined, that is that part of ethics that deals with the study of the interaction between humans and robots.

"The ethics of robotics must try to give an answer to a series of new questions. Is it right that you build robots that perfectly replicate a human being? Is it right that in the future sexual relations between humans and humanoid robots become the rule? Is it right for children to be cared for and educated by a robot? " The situation will become even more complicated with the development of ever more complex artificial intelligences to hypothesize that can develop an awareness of itself, at that point we could consider robots with the same perception as today? For the time being roboethics roughly follows the universal principles of man, so robots that follow the rules of roboethics will not have to discriminate individuals, respect their rights such as privacy, social responsibility, etc.

2.5 Toy robot currently on the market

In the following, a review of several toy robots currently on the market is provided.

2.5.1 Sphero Sprk+

Figure 7: Sphero Sprk+

Sphero Sprk+ (Figure 7) is a programmable polycarbonate robot ball that can reach seven kilometers per hour. It is possible control and program Sphero with a smartphone or a tablet, connecting them using Bluetooth devices, Sphero can also customized with a great number of accessories .

Sprk is an acronym: the letters that compose it are the initials of School, Parents, Robots, Kids. The acronym means the different uses for the robot, indeed it could be used in school or robotic courses, it is a toy robot it can be played in your home and it is provided with a SDK so it is full programmable and it could be used in great number of applications like learning maths

science and writing program

.

2.5.2 Ollie

Figure 8: Ollie

Ollie (Figure 8) is a cylindrical robot capable of reaching speeds of 20 km/h, controlled with an application on both Android and IOS operating systems with a range of over 30 meters. The application gives the possibility to control

everything about the robot from the speed to the acceleration, moreover it is possible through dedicated commands to perform tricks with the robot. Even Ollie is customizable and it is possible to replace the tires with some more performing ones, thus allowing the robot to be able to run also on ramps, moreover it is provided with different coloured LEDs. The robot is totally reprogrammable making it suitable also for educational purposes, in fact it can be used both in school courses but also in university courses.

2.5.3 Vector

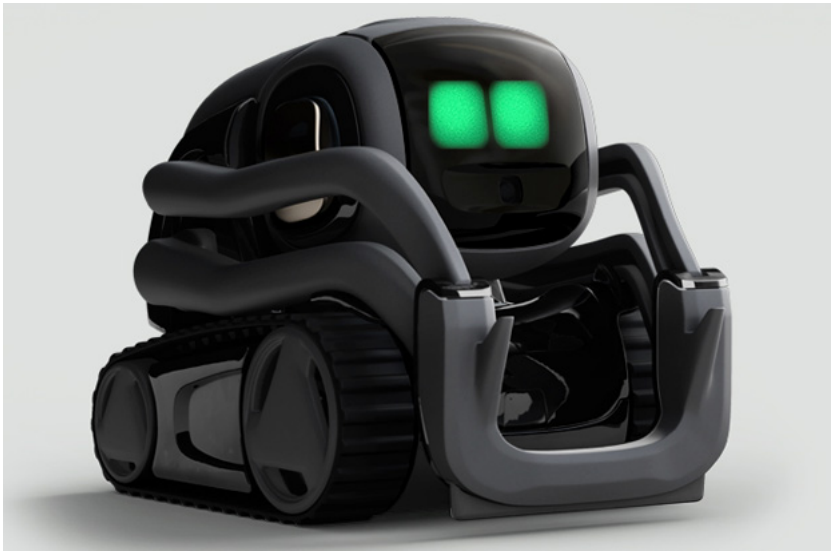

Figure 9: Vector

Vector (Figure 9) represents in some ways the evolution of Cozmo the robot used in the development of this thesis. Users will no longer need to rely on an external application to interact with the robot but can interact using voice commands to provide information such as weather forecasts or any data available online. It moves independently, identifying any obstacles, to

get around them whenever possible. In addition, different expressions appear on his hi-tech face depending on mood.

2.6 Robogame strategy

In recent times the explosion of computing power in microprocessors, the development of new artificial intelligence algorithms and, more generally, the miniaturization of technology, has led to a significant development in the toy robot industry and in entertainment. From these factors several video game companies have developed a new paradigm with which to sell new products, in the new way of developing video games, the limit of the screen is eliminated, so that the player can interact more naturally with the gaming environment, moving taking objects around you or using your voice in some cases. Obviously this leads to disadvantages such as a different setup for each game by increasing the price of the product.

It seems a natural evolution to eliminate the screens to give the player the opportunity to physically interact with the game system. This new paradigm is called PIRGs Physically Interactive RoboGames, and its goal is to bring the game into the real world by inserting, in the context of playing, a robot either as an opponent or as a player team. The main aspect of the PIRGs is to offer a new type of entertainment to the public.

One of the fundamental things of the PIRgs is that the environment reacts in a rational manner to the player's actions and that the robot is a completely autonomous agent capable also of unexpected behaviors of the player's point of view.

It has been observed that it is very important to program the artificial intelligence that controls the robot so that it is as close as possible to the player. Too much artificial intelligence will lead the player to tire of the game early, not offering them enough stimulus and a challenge, while on the other hand, too aggressive artificial intelligence could generate a sense of frustration in the player that could abandon the game prematurely. . It is therefore important to balance these two characteristics as much as possible so that the robot is as equal as possible to the player, especially in the case of the robot playing as an antagonist.

2.7 PIRGs and Phygital play

As said, PIRGs represent category of games built around a robot, which is put in a context peer to the human player,with an autonomous and rational behaviour, so that the player perceives the robot as an intelligent part of the system.

Phygital play in turn, is a contamination of physical and digital elements in games that could contribute at increasing the level of entertainment,by bringing gaming back to its primordial roots and fostering healthy and social behaviours (e.g., by reducing time players spend in front of a screen) [14]. It has been demonstrated that the combination of these two principles provides a new vitality to the gaming experience, in particular robots born with a specific role find new life in this game context. In conclusion,the tendency to bring the game back to a real environment, combining digital tools and real objects can be ascribed to the term phygital. IS' It is possible to assume some

basic characteristics in the creation of a "phygital" scenario. In this work a projected environment was used to transform a table into a game surface with which players can interact through the user interface an example is shown in Figure 10.

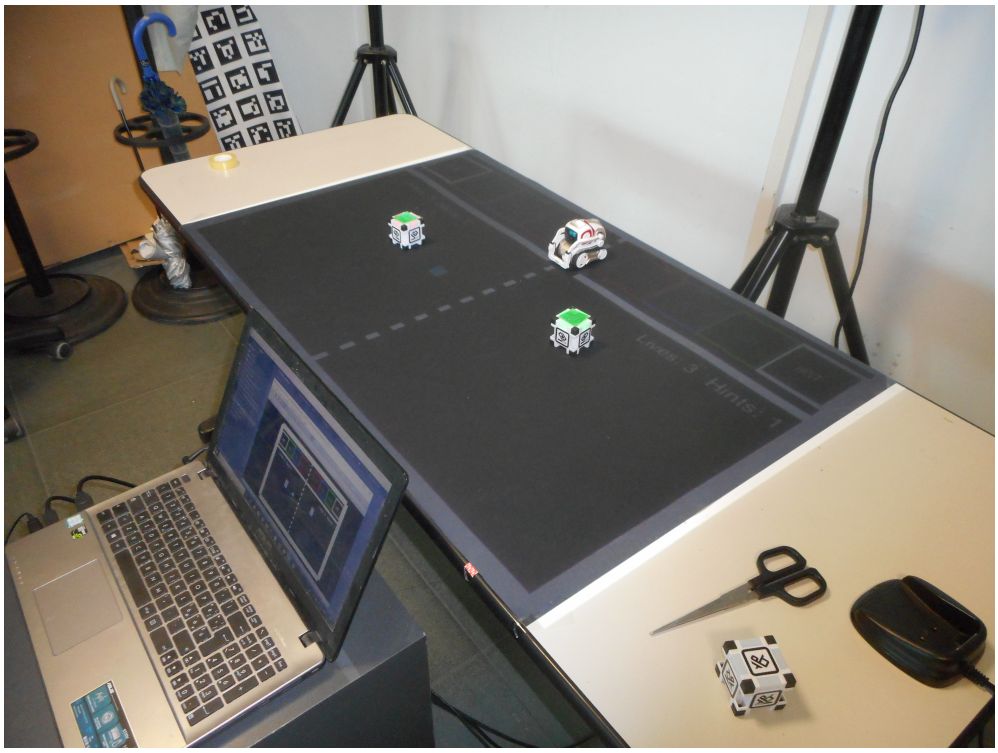

Figure 10: Example of setup of the game implemented in this thesis

"In the frame of robotic gaming, PIRGs are games in which autonomous agents (either digital or physical, including at least one robot and one human player) interact in a possibly unpredictable environment according to determined rules, with the ultimate goal of letting players have fun" [14]. Several guidelines have been set for suggesting game designers how to create effective PIRGs:

- consider environment's characteristics and agents' capabilities;

- define a story;
- reduce, reuse and recycle existing software and hardware components;
- foster safety and simplicity;
- make the robot appear as a rational agent;
- exploit senses.

2.8 Previous work

In this section,previous works will that inspired this thesis will be presented. These works show an architecture similar to that used in this work, will have the same paradigm of play, the PIRGs even if the objectives and the robots used may differ.

2.8.1 Jedi training

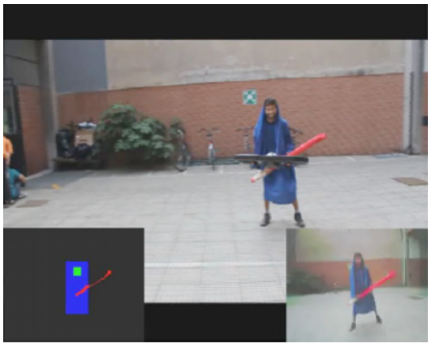

Figure 11: Jedi training

One of the first game that follows PIRGs paradigm is Jedy Training (figure 11). Indeed, to define the guidelines for the robotic game, the work also

illustrates one solution and strategies to use a drone equipped with a camera and a physical interface. The game incorporates all the guidelines defined by the PIRGs:

The game wants to recreate the training of Luke Skywalker at the edge of the Millennium Falcon as in Episode 4 of the famous Star Wars saga,: In the movie Luke was training to parry the laser beams with his own lightsaber with the difficulty of being blindfolded, this to refine his bond with the Force, a mystical element that belongs to the Star Wars saga.

In the designed game instead a drone flies around the player while shooting laser beams that he will have to avoid or parry with the sword laser. If the beam hits the player, the drone gains a point, vice versa it will be the user to earn a point. In a scenario of higher difficulty, it should be possible play, as in the film, without using eye contact but his own hearing. The game's requirement is a fairly free space to allow the drone to move around the player at a distance unreachable with the lightsaber that is the physical interface in the game system.

2.8.2 Protect the treasure

Figure 12: Protect treasure

The concept of Protect the treasure is based on an attack and defence game. The playing area is projected onto the ground, Figure 12). The robot in the role of attacker will try to collect the treasures, virtual objects projected in the playing field, passing over it, the player in the role of defender will have to prevent the robot from collecting all the treasures by placing his body between the robot and the treasure, the body is represented by a red circle that acts as a virtual counterpart, if the robot manages to collect all the treasures it wins. This represents a mixed reality game where the user interface is the player's own body.

2.8.3 Robot ARena

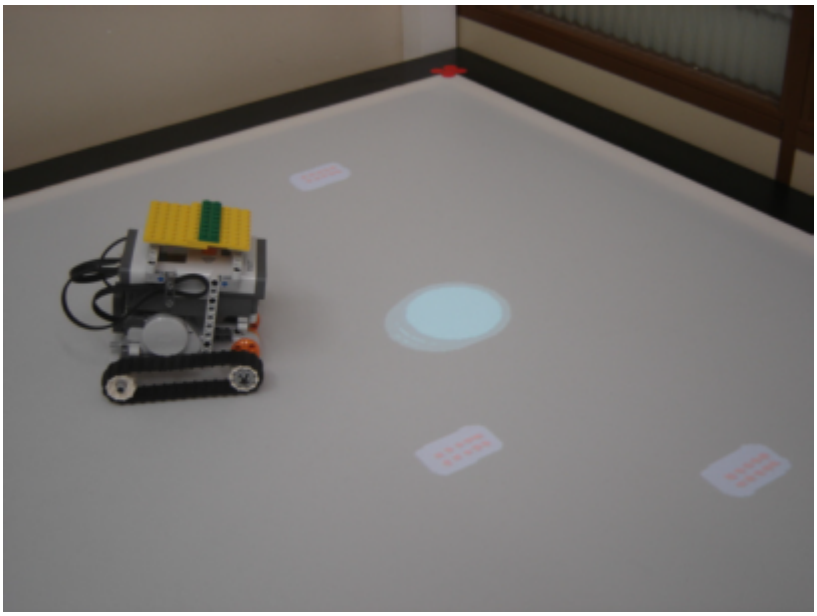

Figure 13: Robot ARena

Robot ARena proposes an augmented reality platform for game development robots. The architecture of system is composed of a projector with a 45 inclined mirror, a table of projection, a camera and a robot (Figure 13). The software infrastructure is divided into four subsystems, each responsible for a specific part: visualization, tracking, real elements, communication. The robot is also in this case a commercial model, assembled using Lego NXT. The system offers two game prototypes, FootBot ARena and TankSpace. In the first, two robots, one virtual and one real, play a football game with a virtual ball. Some collectible items are added to help the player (bomb, beam, shield), other obstacles can obstruct both the robot and the ball. The robot is completely in this case autonomous, as the game in question is designed to explore interaction between real and virtual. In the second

game, the main objective is instead to explore the use of tangible interfaces, with the possibility of controlling objects that interact and interact with the projected content. The game is set in space and the player controls a spaceship that must destroy enemy ships. To do this, the player can control the aim of the cannon by moving a physical object in the arena of play. The item it must be aligned with the corresponding projected shape to achieve success. There spaceship in the prototype is also virtual, although a possible conversion is expected in a robotic element; therefore, at the moment the game considers the interaction between player and virtual environment, but still lacks interaction with the robot.

2.8.4 Protoman Revenge

Figure 14: Protoman Revenge

The designed game proposed a duel between a human player and a flying drone,(Figure 14) Both have the ability to shoot a laser beam. Moreover the player has the ability yo defend himself/herself with an equipped shield. The player must hit the drone with is gun without getting hit in turn. If the player remains with no lives he loses the game.

3 Technologies

During the development of the thesis, different technologies are used both hardware and software, following they will be presented all the technologies used, to give an overview of the work done.

3.1 Unity

The game filed and the physics of environment are developed in Unity editor. Unity is a cross-platform game engine developed by Unity Technologies. Developed for the first time in 2005, Unity was continuously expanded year after year until it came to a very versatile video game design editor. In unity, in fact, there is already a graphic and physical engine that allows rapid prototyping of projects, and an easy to use implementation. Unity allows creating both 3D and 2D layouts for game development.

In this project was chosen 2D layout to developed game engine.

3.2 Python

Python is powerful... and fast; plays well with others; runs everywhere; is friendly and easy to learn; is Open. [1].

The robot logic and tracking were developed in Python. Python is an high-level, general-purpose programming language, it is created in 1991 by Guido van Rossum, it is though to be a simple but complete language program,indeed it is often described as "batteries included" language due to a great number of library that could be implemented in your code.

Specifically, the libraries imported into the Python environment are:

- Cozmo's library, necessary to use SDK in python environment;
- Time library, used to handle time-related tasks;
- Zmq library, used to implemnt socket,it will be discussed in more detail later;
- Random library, used to generate random numbers,it is needed to generate random sequence of colours;
- Threading library,it is needed to implement threads, in this kind of work it is fundamental works with parallel tasks;
- OpenCV library, a powerful computer vision library,used to perform collour detection algorithm,it will discussed more in detail later;
- Ntkinect library,a wrapper that allows to import rgb frame from kinect in Python environment.

3.3 ZeroMQ

ZeroMQ (also spelled ØMQ, 0MQ or ZMQ) t is a synchronous messaging library created with the purpose of communicating different processes, with the APIs provided it is possible to create sockets quickly and easily. It is used to established the communication between the two process in python and in unity, the API provides different patterns of communication; in this thesis is used reply request pattern, were the process in unity, as client, asks

information from python process, as server,like the position of cubes, the game status and so on.

3.4 OpenCV

OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms [3]. It is used to implement the tracking algorithm using colour detection and contours analysis.

The principal integrated functions that have been implemented in the thesis are:

- $cv2.cvtColor(inputimage, flag)$,a function that allows to change the colour space, ()Figure 15),where flag determines the type of conversion.

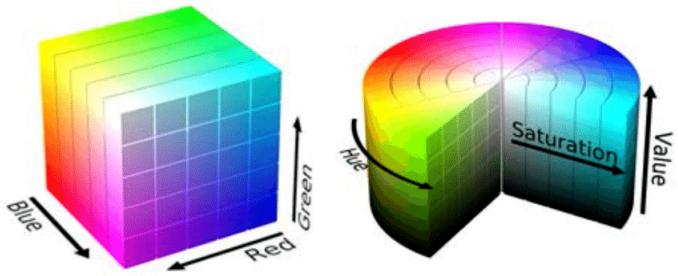

Figure 15: Change of colorspace from rgb to hsv

- $cv2.getPerspectiveTransform()$,$cv2.warpPerspective()$:These two functions integrated in OpenCV are used respectively to calculate a 3x3 transformation matrix and to apply this matrix to an input image to obtain a prospectively transformed an output image, an example is shown in Figure 16.

These functions have been used to eliminate the error of perspective due to the inclination of the camera with respect to the game plan, it was essential to correct this error for a correct reading of the coordinates provided by the tracking algorithm [8].

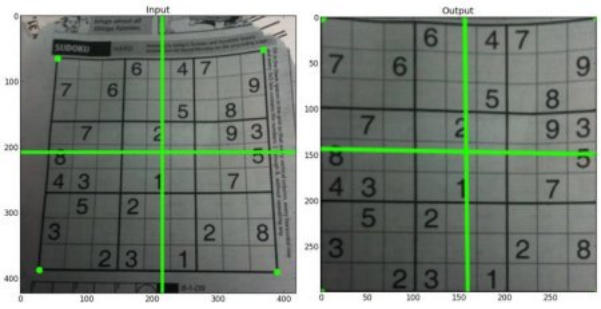

Figure 16: An example of perspective transformation

- $cv2.inRange$, a function the allows to perform a threshold of image in a chosen range, an example is shown in Figure 17

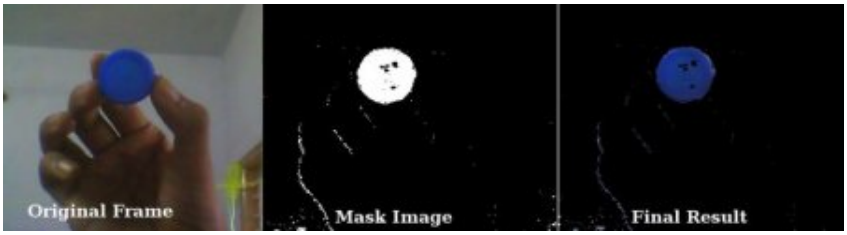

Figure 17: An example of how to extract blue object alone

- $cv2.erode(), cv2.dilate(), cv2.morphologyEx()$: These functions have been used to eliminate the error of perspective due to the inclination of the camera with respect to the game plan, it was essential to correct this error for a correct reading of the coordinates provided by the tracking algorithm.

These three functions were used to implement some morphological transformations on the frames acquired by the RGB camera. Morphological transformations are particular transformations that act on the shape of the image, usually they are used on binary images, so in this thesis they are applied after using the treshold function, an example of erosion e dilatation is shown in Figure 18. The idea at the base is to eliminate all the noises due to the ambient light conditions, both artificial (lamps and the light emitted by the projector) and the natural ones (mainly due to the sun), these could generate blobs, these are nothing more than agglomerates of pixels that could distort or disturb the tracking algorithm.

To achieve this, a special technique called erosion followed by dilation is used. The basic idea is very simple, all the blobs perceived by the image treshold are eroded, including the blob that could be the object one would like to trace, in this way all the smaller blobs simply disappear leaving only the spot of pixels larger than corresponds to our object, then obviously the expansion has reduced the size of our object to be traced, the dilation function is applied which is the opposite of the erosion function to return the object traced to its original dimensions, example in Figure 19.

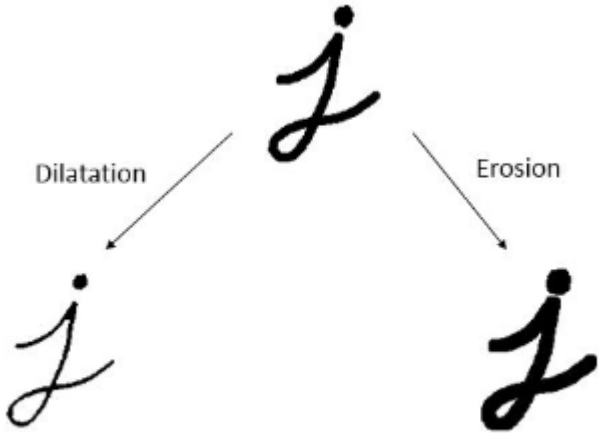

Figure 18: An example of erosion and dilatation

Figure 19: An example of how to use morphological transformations to eliminate noise [11]

3.5 Projector

Figure 20: Optoma GT760 projector

The projector used is Optoma GT760, Figure 20, The projector uses the Digital Light Processing (DLP) technologies to project a high quality imagines necessary to implement a robotic game. It have also an incorporate loud-speaker but is not used in this work.

In this works it is used only a part of projection area, that is put in evidence by a black cardboard with dimensions 100x70 cm put on a table, since the small dimension of the robot suggests a little scale game.

3.6 Microsoft Kinect V2

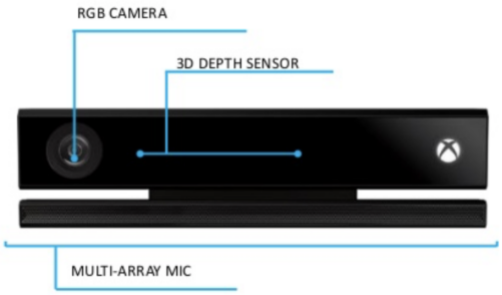

Figure 21: Kinect V2

The Kinect (Figure 21) allows to interact with the game using the body, based on the use of gesture and voice. The Kinect is involved in a lot of different fields from computer vision robotic and biomedical applications. After the launch of SDK on June 2016, a large community is formed, permitting to use Kinect not only as game devices but also a measurement system. Kinect, using a RGB camera, an IR emitter and an IR camera, can acquire a depth and visual imagines extending these technologies to low cost project.

However The Kinect used in this thesis is Kinect for Windows 2.0 (hence the code, V2) especially developed for Windows operating system. The SDK are available from summer 2014,enables developers to create applications that support gesture and voice recognition, using Kinect V2 sensor technology on computers running Windows 8,The Kinect for Windows Software Development Kit (SDK) 2.0 Windows 8.1, and Windows Embedded Standard 8. In this work it is only used RGB camera, employed to perform cubes tracking by colour, without support from official SDK but using OpenCV library. Kinect V2 can acquire RGB frames at resolutions of 1920x1080 and GBRA format at a speed of 30 fps in case of good visibility conditions, otherwise it drops 15 fps to give more exposure time to the camera.

Since the place and resolution of each sensor is different, the data is obtained as a value expressed in the coordinate system of each sensor. When using data obtained from different sensors at the same time, it is necessary to convert the coordinates to match.

Kinect V2 has 3 coordinate systems, ColorSpace, DepthSpace, and CameraSpace, (Figure 22). There are 3 data types ColorSpacePoint, DepthSpace-

Point, and CameraSpacePoint representing coordinates in each coordinate system [12].

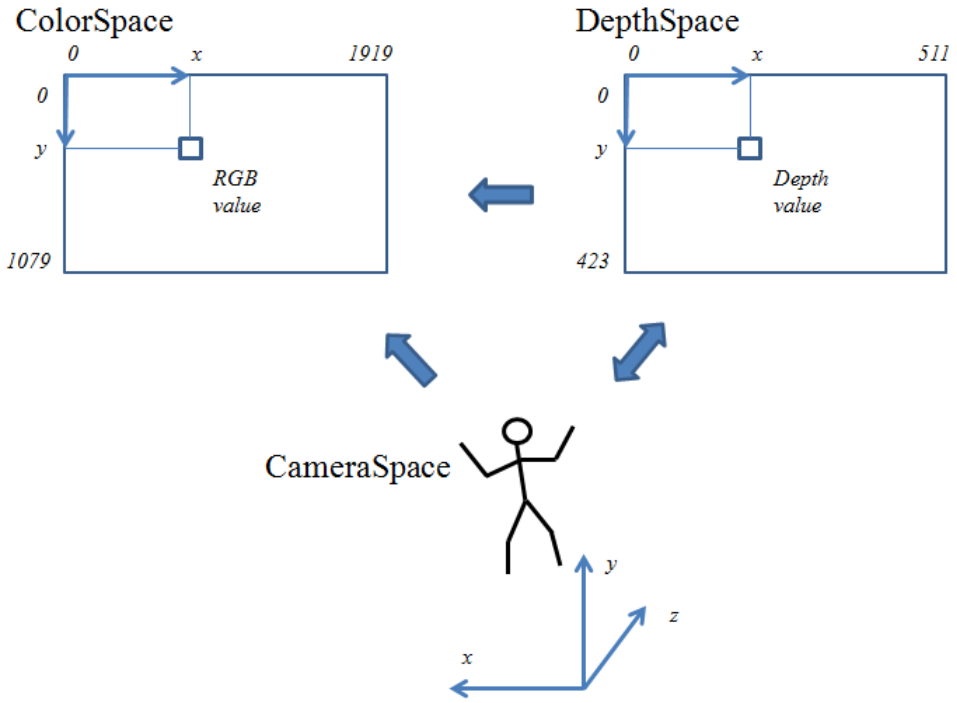

Figure 22: Kinect V2, different coordinates spaces [12]

3.7 Cozmo

Figure 23: Cozmo robot

As said, Cozmo is a palm-sized robot developed by Anki with personality inspired by animation film wall-E [6](shown in Figure 23), is a smartphone controlled robot, that means to interact with robot or launch SDK model you need a smartphone with installed the official application still created by Anki .

Cozmo was designed to interact with both the external environment and the people around him through a display that shows two large blue eyes and through sounds. Thanks to these two instruments, Cozmo is capable of great expressiveness making it look more like a pet than a toy robot. Anki has in fact hired a Pixar animator to participate in the creation of Cozmo, the result is a robot with an extraordinarily similar personality to the WALL-E robot from the famous animated film [7].

Cozmo's built-in camera also lets it remember faces and recite names. It comes with three cubes that carry sensors and lighting that are used in various games and also let Cozmo plan paths. Cozmo has several sensors:

- camera;
- cliff sensor;
- gyro;
- accelerometer.

Each of these sensors can be accessed through the dedicated SDK. Cozmo also has an advanced location system based on visual markers and the odometry of the robot, where the origin is initial Place of the robot and is reset every time the robot is moved or shaken.

Included with the robot are also sold three cubes, each including an accelerometer, Cozmo is able to recognize them through the integrated camera (each cube has a unique identifier), and to save their Place in an integrated map.

The user can either interact directly or with the robot or through its three cubes, Cozmo connects to its cubes through radio waves and is able to understand if and which cube is tapped or shaken. This type of interaction already offers multiple possibilities to create games only with the hardware present in Cozmo; Anki through the application already gives the possibility to challenge Cozmo in pre-integrated games(like in Figure 24), ranging from races of reactivity, memory or quizzes to prizes.

Figure 24: Cozmo plays against a human player

There is also the possibility of creating your own games and downloading the games created by the user community through the application, in this way a considerable longevity of the product is guaranteed both for users who do not wish to program and for those who want to try to program a robot already having many tools available.

4 Design

In this work the difficulty encountered in devising the game was to combine the characteristics of a robotic game with a projected environment, that is to integrate well the possibilities offered by the virtual environment, putting the robot as protagonist of the action.

The original idea was based on the famous game Pong. Players using the cubes as physical objects must hit a virtual ball, but unlike the original game players do not get points by scoring on the opposing side of the field, but they must hit the robot, that it moves to the centre of the game's field randomly, in a specific area, for example its back.

In this version the condition of interaction between physical and digital was respected, but after a few tests it was immediately noticed that the role of the robot was not predominant in the game since the players were too concentrated on the ball and did not perceive the robot as a fundamental part of the game. It was therefore decided to change the paradigm of the game: instead of putting the robot as physical goal of the game, it was put in the role of active arbiter, after which the rules of the game implemented in the thesis will be explained in detail.

4.1 Game's rules

At the beginning of the game the robot is placed in the upper part in the center of the field, while the two players take their seats at both ends, each of the two players holding a cube, this will be used as a racket to hit the

virtual ball.

The game is divided into rounds; at the beginning of each round the robot moves to the center of the field and speaks a series of two colours, randomly choosing between green, red and blue.

Once the series is said the robot returns to the end of the playing field and will start the challenge, at this point a virtual ball materializes in the center of the field and it will start to move randomly or towards the player on the right or towards the player to the left.

Players must intercept the ball by appropriately setting the LEDs of the cubes following the series of colours enunciated by Cozmo; for example if Cozmo says red and blue, the first player to receive the ball must set his red cube and intercept the ball, while the second player must instead hit the ball with the blue cube. The fact that ball can move both to the right and to the left at beginning of each round, forces players to memorize the entire sequence of colours instead of only half. To increase the robot's participation even during the game, the players are given the opportunity to ask for a suggestion from the robot, when a player asks for help the ball stops, the robot indicates to the player the exact colour of the sequence, player sets the right colour and the ball moves again, each player during the game has five hints that he can spend at will.

Figure 25: The players are listening to colours sequence pronounced by the robot

The objective of the game is to test how the interaction with tangible objects modifies the perception of the game and the robot. For this reason three different versions of the same game have been developed, in which it changes how the players choose colours for their cubes and as how one asks for the help of the robot; a different playing field was designed for each version, each functional for the version implemented.

4.2 Place version

In this version, specific areas for each player have been designed on the playing field (Figure 26). The player to ask for help or change the colour

must simply move the cube to the correct playing area.

In the case of a suggestion request, the robot moves in front of the correct colour area, says the colour and returns to its place.

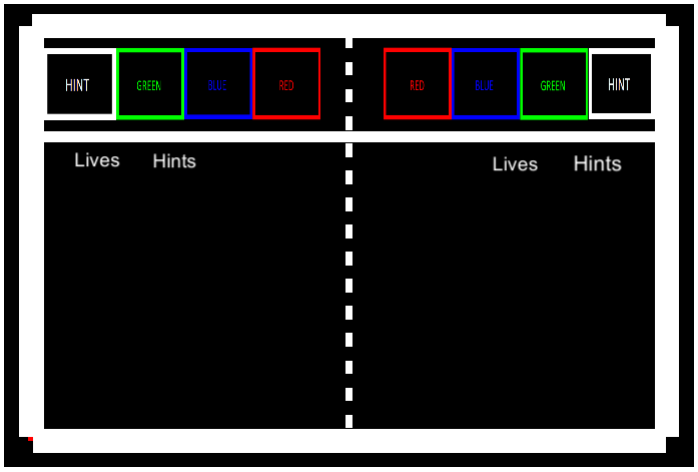

Figure 26: Place version

4.3 Tap version

In this version the player to change the stage colour on the cube once and to ask for the suggestion stop twice consecutively, the playfield is shown in Figure 27 .

In the case of a request for help, the robot turns to the specially designed virtual cube, says the correct colour and simulates tapping the virtual cube, the cube lights up with the correct colour to indicate to the player the exact colour to be set.

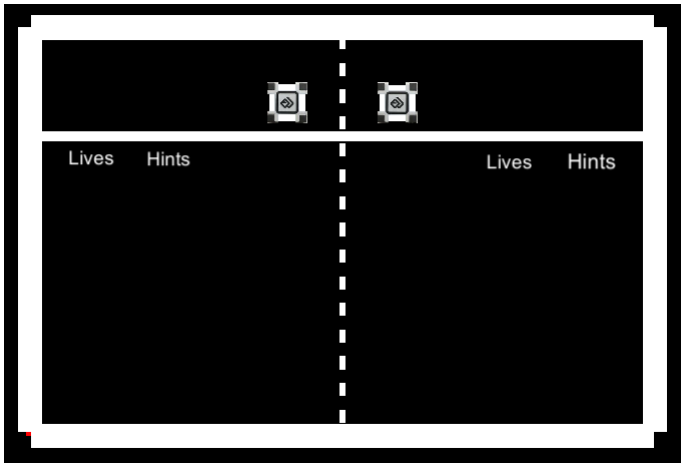

Figure 27: Tap version

4.4 Mid-air version

In this version the player to change the colour brings the cube in mid-air always inside the playing area shown in Figure 28, reverses the top of the cube with a rotation of at least ninety degrees and then brings it back to the original Place, the cube in this way change colour; to ask for help the player instead vigorously shakes the cube along its vertical axis.

In the case of a request for help, the robot turns to the player and simply tells the player the right colour.

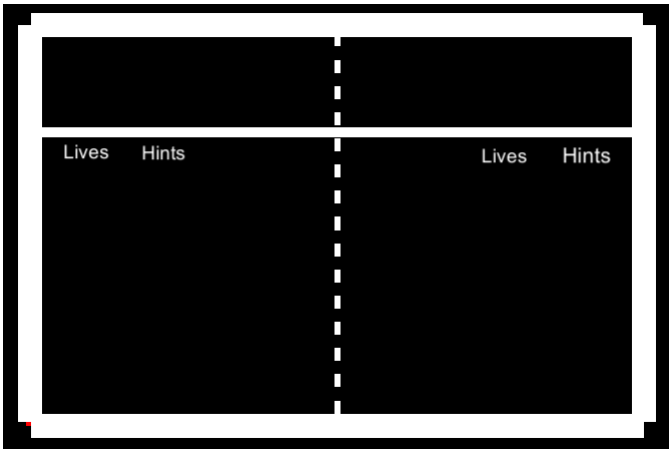

Figure 28: Mid-air version

5 Implementation

In the implementation of the game a computer was therefore used to manage both the unity environment and the logic of the robot. A mobile phone with Android operating system connected to the computer via USB and to the Robot via Wifi; the smartphone serves as a bridge between the robot and the Python code that runs on the computer which uses the robot's SDKs.

The projection and the tracking are performed from above with respect to the playing area, and therefore it is necessary to Place the projector and the Kinect in an elevated place with respect to the playing surface, at the centre of the projection area was placed a table with above it a black card measuring one meter by seventy to better define the playing field. Afterwards, the playing area is projected onto the black cardboard so that they are overlapped.

As already mentioned, a projector, a Kinect and the Cozmo robot were also used. As already mentioned, the game was developed in three different versions, each of these has its own Python code and scene in Unity.

The first problem to be solved is the integration of the different technologies used, in fact the robot SDKs are written in Python language, while the Kinect and unity have C# as their main language. It was therefore decided to create two different scripts, one in Python code that manages the logic of the robot and tracking using the Kinect (with the help of the Opencv library) and the second script in the Unity environment that manages the game engine and the graphic part.

For the communication between the two processes it was decided to use a

socket. The library used is ZeroMQ. To start the application it is necessary to manually start the Unity and the Python processes.

starts, the following steps are executed:

- The socket is created on the Python side, which is bound in a certain channel,the socket waits until it receives the order from Unity.
- When the application starts in Unity another socket is created, that communicates with the Python process, when the connection is successful established, Unity sends the start command to Python, after which it opens two threads, a secondary thread management socket, and a main one that takes care of all the integrated functions in Unity and the game code.
- When the process written in Python receives the start command from Unity, it opens three threads each with a specific role, the first deals with the logic of the robot itself, the second manages tracking through the Kinect while the last one is the thread that handles communication with Unity via the socket.
- During the course of the game all the threads run independently of one another, the communication between the threads was managed through the global variables both in Unity and in Python.
- When the game ends, the event is recorded from the Unity environment, which before terminating the process and closing the threads sends a notification to Python, which in turn will close all the threads and terminate the application correctly.

5.1 General architecture

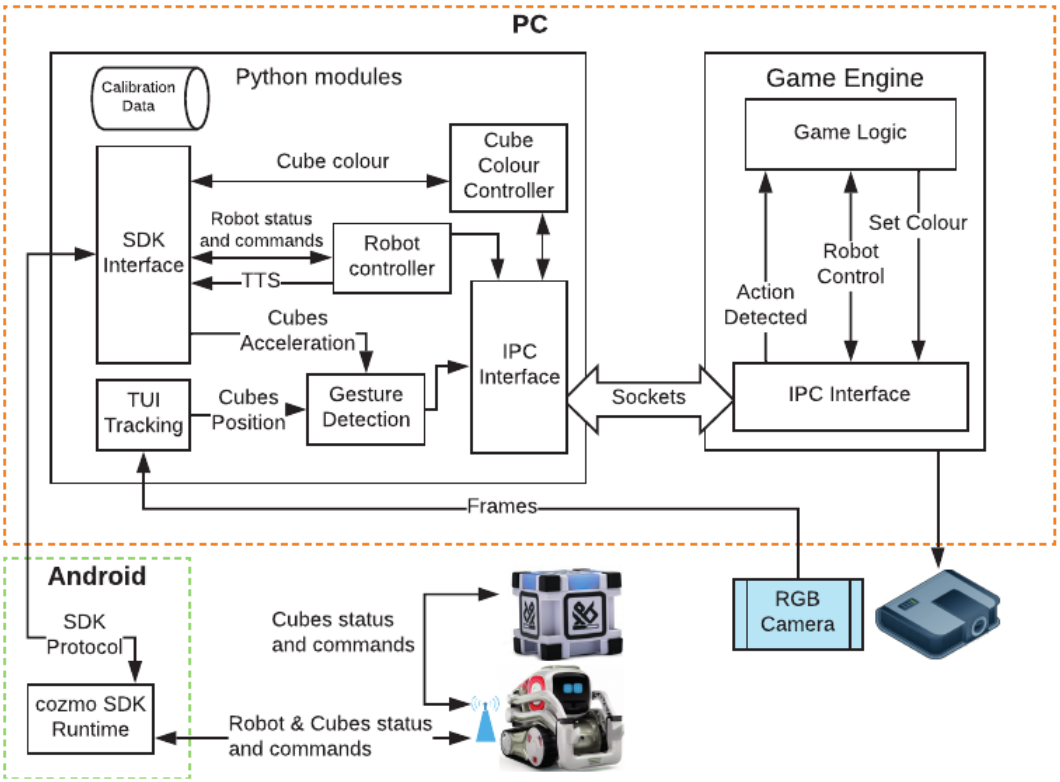

Figure 29: Architecture of the gaming system

The setup included Cozmo, two cubes, a RGB camera, a projector, an Android smartphone and a PC running Windows 10.The projector was mounted near the ceiling in order to project the image on the table from the top. To improve the quality of the projected image, the table was covered using a black cardboard of size 100 x 65cm, which is also the size of the play surface. In the immediate nearby of the projector it was mounted a Microsoft Kinect with the same orientation.

The setup of the game system is shown in Figure 30.

From the game architecture observed in the Figure 29 we can note three

main elements:

The first is the Android operating system which, connected to the computer; it has the task of receiving and executing data commands to the Robot. In fact the SDK protocol that allows the robot to be controlled is implemented in the application developed by Anki. The cubes are connected through radio waves to Cozmo itself.

In the Python block, the program that controls the robot and the program that manages the control of the cubes is written, such as the colour and all the events that affect the activity of the cubes in the system. In the Python module, the TUI, Tracking User Interface is also implemented. , which is the algorithm that deals with the tracking and location of the cubes in the game field.

All the sub-modules converge in the IPC Inter Process Communication, implemented with the use of a socket. The IPC is used to connect the Python module with the game engine implemented in Unity. The game engine takes care of all the virtual elements and of the physics and management of the game logic.

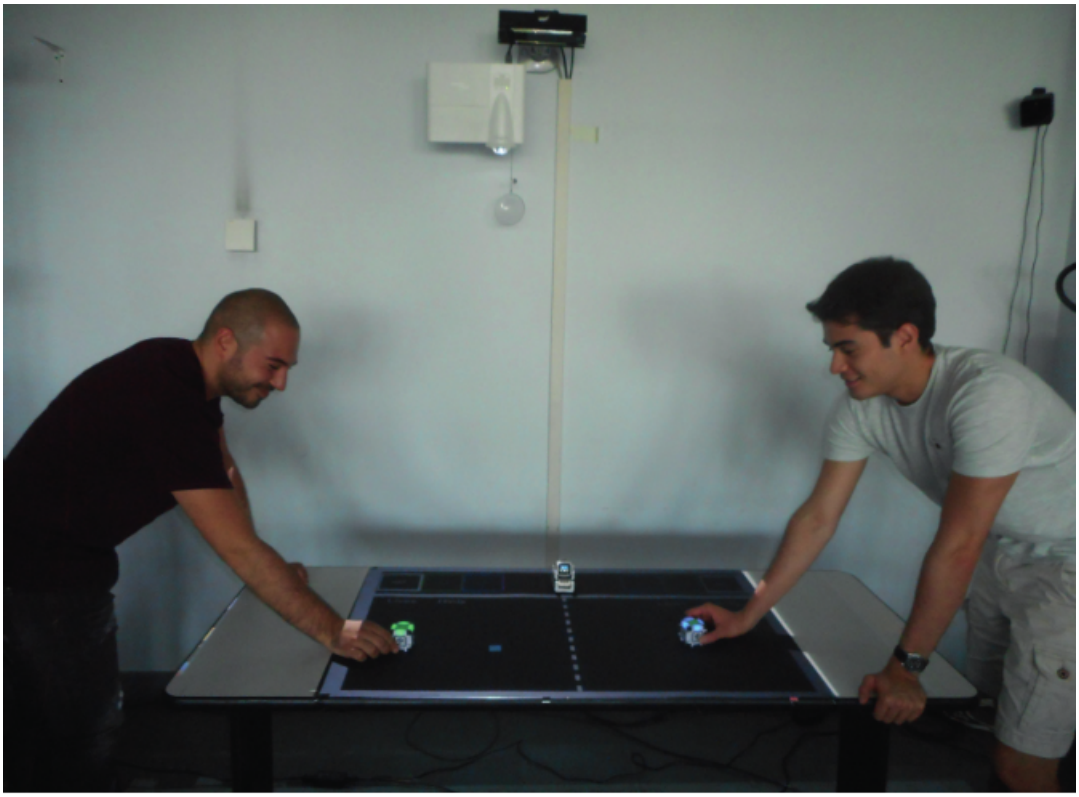

Figure 30: Setup of the gaming system

5.2 Tracking

The tracking algorithm was implemented in Python with the help of the OpenCV library. To extract data from the Kinect in Python environment it is used the NtKinect wrapper. Each frame acquired by Kinect (Figure 31) is transformed into a numpy array, a data structure compatible with the OpenCV library. The following operations are performed for each frame acquired.

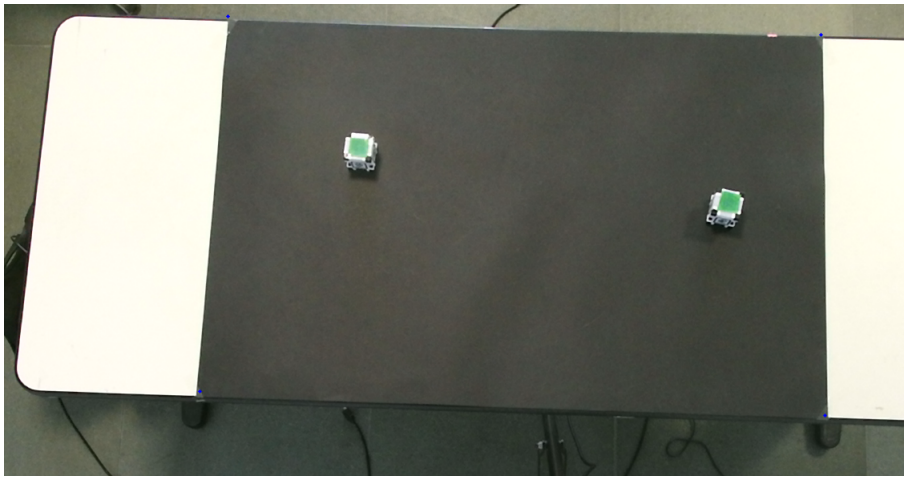

Figure 31: Frame acquired by Kinect

- A perspective transformation is applied to select the playing field and eliminate any errors of perspective due to the not exactly perpendicular placeing of the Kinect with respect to the playing surface, to carry out the transformation it is necessary to select 4 points corresponding to the corners of the playing field in the image of inputs, of which 3 must not be collinear. The output frame is shown in Figure 32.

- At this stage a transformation matrix is calculated and it is applied to each point of the game surface. The output image that represents the playing field is then processed.

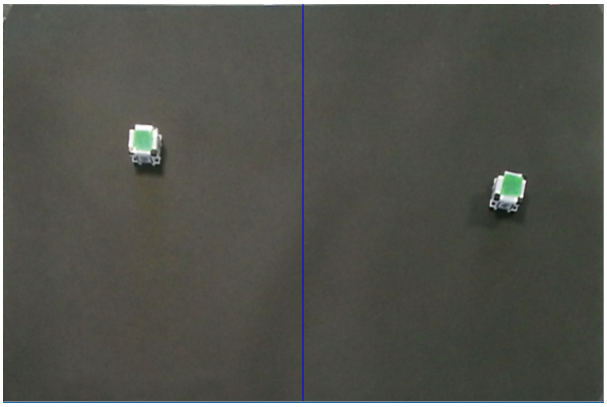

Figure 32: Perspective transformation

- The playground is obtained, divided in two ROI (region of interest) which correspond to the two portions of the field destined to the players. In each portion of the field the tracking algorithm is applied.
- A change of colorspace from RGB to HSV is performed, recommended to find the correct range of values more easily.
- A treshold of the HSV image is performed in the green range.
- A morphological transformation of erosion is carried out followed by dilation to eliminate any noise. The result is shown in Figure 33.

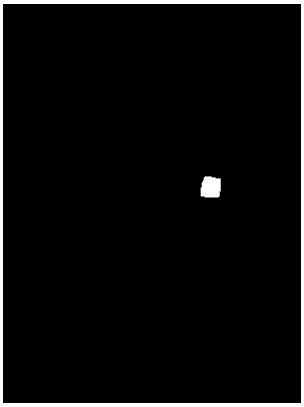

Figure 33: Final mask

- The object is extracted and the countour is traced (Figure 34). Among the possible countours, the one with the largest area is chosen. The center of countour that corresponds to the center of traced cube is then found. Each coordinate of the cube must be scaled so that it can be transferred correctly into the coordinate space in Unity.

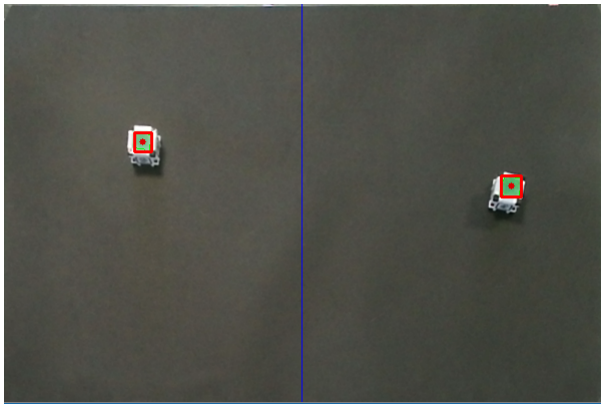

Figure 34: The cubes are tracked

- To do this during the setup phase the projection of the playing field was made to coincide with the black card, which represents our physical play area. The unity of origin placed in the corner was made to coincide

physically lower left with the corner of the black card always placed in the left corner, which also represents the origin of the coordinates in the space of the frame. Once the two origins have coincided, it is sufficient to specifically scale the coordinates obtained from the tracing according to the equation:

$$X_u = X_{img} * X_f \tag{1}$$

$$Y_u = Y_{img} * Y_f \tag{2}$$

In the above equations, X_f and Y_f are the scale factors obtained:

$$X_f = Width_u / Width_{img} \tag{3}$$

$$Y_f = Height_u / Height_{img} \tag{4}$$

Then the ratio of the quantities in Unity and in the frame is calculated. At this point the coordinates are sent to Unity through the socket, which are used to move two virtual objects (the counterparts of the cubes) in relation to the displacements of the real cubes.

It was decided not to trace the robot because it has a sufficiently precise autonomous localization system. In fact the robot sets its origin at the starting point and sets its own Cartesian plane; the robot will therefore move

coherently with its own reference system, for this reason it is important to place the robot correctly in its origin point.

5.3 Cozmo's thread

The intelligence of the robot is implemented in a new thread using dedicated SDKs. The program performs the following actions.

- Cozmo acquires the cubes through the appropriate command provided by the SDK, Cozmo and the cubes communicate via radio waves, reads the identifiers of each cube and sets the event handlers with their respective callback functions

- Cozmo enters a cycle that has exit condition the end of the game. Always through the functions provided by the SDK Cozmo instructs the players that a new game has started, with a function implemented in Python, it randomly generates two colours between red blue and green and then it communicates to players the chosen colours.

- Cozmo enters a further nested cycle that has as its exit condition the end of each round, in this cycle Cozmo cross-checks the information shared with Unity.

- In this cycle Cozmo checks each bounce of the ball on the cube (information that comes from unity) and checks that the cube is setted with the correct colour. It also checks the event "Suggestion" required by the players".

- Depending on how the round ends, that means if the players have successfully completed the entire colour sequence or a player has made a mistake, Cozmo gives different feedback to players: it could be cheerful / happy if the players have completed the round correctly, or sad / angry if the players have made a mistake.

- At this moment Cozmo checks the lives of the players: in case one of the two players has ended the lives, Cozmo announces the winner and the game ends, otherwise Cozmo will start another round adding two more colours.

5.4 Gestures

As already stated, three different versions have been implemented. Each version differs from the other mainly in how the player changes colour to his cube and how the robot provides hints.

5.4.1 Place version

In this version the player changes colour and asks for help placing the cube in a specific play area. For each area designated in Unity, a box collider was created. When the virtual cube enters the collider box, Unity calls a function that sends information through the socket to Python code. When that changes the designated cube to the correct colour. An example is shown in Figure 35.

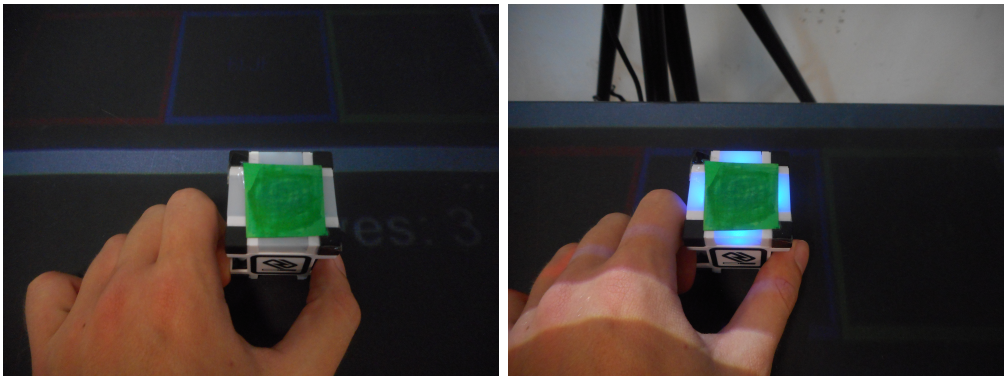

Figure 35: An example how to change colour in Place version

5.4.2 Tap version

In the Tap version, the Python code when a cube is tapped, it calls a function that triggers a delay of three tenths of a second ; if during this time another tap event is received, it is perceived as a suggestion request, otherwise the cube will simply change colour. An example is shown in Figure 36. One more condition has been added to prevent false positive, the condition is that the cube is stopped. The information are provided by the values of accelerometers present in the cubes.

Figure 36: An example how to change colour in Tap version

5.4.3 Mid-air version

For the Mid-air version, it was decided to assign the colour change to the Kinect tracking. It measured the area variation of each cube, if the area of a given cube drops below a certain value (here set empirically to 50) is perceived as an attempt to change colour, once the area of the cube has returned to the standard value (> 200 also empirically measured) the code will change colour, furthermore it is necessary that the cube is in motion to avoid false positives. The example of gesture is shown in Figure 37. For the call of the suggestion, it was refereed to the internal values of the accelerometers of the cube; if the acceleration on the z axis is greater than 100 and the acceleration on the x and y axes are minor than 70, then the suggestion is called (these values have also been found empirically).

Figure 37: An example how to change colour in Mid-air version

5.5 Socket

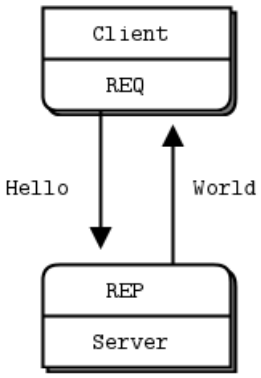

Figure 38: Reply-Request Socket

The socket has been implemented with a reply-request structure as shown in the Figure 38, where the Unity client asks for the status of the robot and tracking to the Python server. The communication protocol was implemented in a way that the processes could exchange strings with variable number of characters, Every Unity request expects a precise type of response. The socket runs in a dedicated thread because it is a self-blocking system, which is why it was important to implement a threads structure so that other processes could continue to run even when the socket is waiting for the response. The reply-request pattern was chosen because it allowed the implementation of a single socket, since it allows a bidirectional communication between the two processes.

In general the socket has the tasks reported belows.

- It must send the coordinates of the cubes from the tracking system to the Python environment.

- It must send to the Python environment the collision events between the cubes and the ball. they are fundamental for the correct progress of the game,but they are also necessary to permit the system to block the request of suggestion for the last player to hit the ball, thus avoiding further false positives.

- The socket sends the information if the ball starts to move either to the right player or to the left player, this information is necessary as the system must know which of the two players has the permission to request the suggestion.

- The communication protocol foresees a special command that allows to invert the communication: instead of Unity sending a command or request to the Python environment, Unity sends an instruction to the Python environment that requires a state, in this way it is Python which sends a command to the Unity environment.

- Python sends Unity the command to instantiate the ball, it is placed by Unity in the centre of the field, then, when Cozmo has played the "go ball" animation, the socket sends the command to move the ball.

- Python sends the end-of-round status, that is, if the players have successfully completed the entire color sequence or an error has occurred by one of the two players. When Unity received the message it destroys the ball and set up a different animation depending on the positive or negative end of the round.

- Python sends the command to start the suggestion and the end suggestion, activated when a player makes the gesture of requesting help from the robot. When Unity receives the command start suggestion, it blocks the ball, to restart it once the command end suggestion was sent.

- Only for the tap version: Python version, when a suggestion is required from one of the two players, it sends Unity the colour information suggested, Unity instantiate the virtual cube of the exact colour as required by the rules of the game.

- Only for the Place version: This is the only version where the colour information chosen for the cubes does not come from the Python Environment but from Unity, since, as already specified in the rules,different zones have been created in the virtual environment , these are provided with a collider box that when triggered Unity sends the command to Pyton through the socket, which will light the cube with the chosen colour.

- The last command that Unity receives from Python is the end game command. It is called when the lives of one of the two players reach zero, Unity before ending each process send the end command to Python, which once received , it closes all threads opened using a flag variable.

5.6 Game's elements

The elements of the game are all those elements that have been created or used to implement the thesis work, as the "Phygital" paradigm teaches, they are both virtual and physical, they must interact intelligently with each other to create the sense of control that the player has over the digital environment.

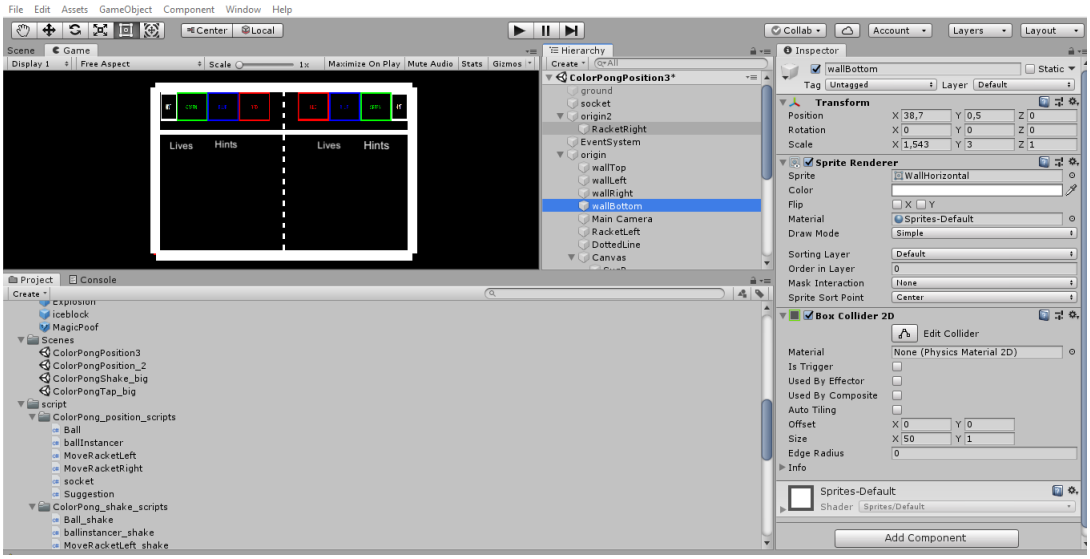

Figure 39: Unity editor interface

The physical objects in the game are the cubes and the robot, the digital objects are all the objects that were created in the Unity environment as shown in the Figure 39.

5.6.1 Socket object

The socket in Unity object was created to receive informations from the python environment and to sort theme to all other objects in the unity environment, it is not perceived as a "tangible" object. In fact, among its

properties it only has the C# script that defines the socket. The script has a method that instantiated a new thread using the awake function integrated in Unity, and when the application starts it launches a method that starts the thread, when the object is destroyed, that happens at the end of the game, the program sends to python the exit command and correctly closes the thread.

5.6.2 Walls

Figure 40: Wall sprite

The walls (Figure 40) are the simplest elements of the game but not for this reason less fundamental. In fact they allow to delimit the playing field imposing to the ball to exit from the designed area. The attributes that they possess are the Sprite renderer that allows the object to be visible in the field of play, and a non-triggered box collider, which means, according to the Unity logic, that it acts as a physical object.It follows the laws of physics already implemented in the unity graphics engine.

5.6.3 Dotted-line

Figure 41: Dotted-line sprite

The dotted (Figure 41) line has a simple sprite renderer. Indeed its role is simply to provide a visual aid to the players to divide the field, it is simply an image and does not interact in any way with the other objects.

5.6.4 Ball

Figure 42: Ball sprite

The ball in unity is defined as prefab (Figure 42). Which according to the unity logic is an object that is instated on demand, is therefore not present at the start of the application but is created when called and subsequently destroyed when it has terminated its task. A cube shape was chosen to follow the idea of the original 80s pong game. The attributes of the ball are:

- The sprite renderer has the function to make the ball visible in the playing field.

- The box collider that allows the ball to react with the other objects in the game. The box collider also has a material that has been called for simplicity "ball material" useful for controlling the bounce factor of the ball.

- The rigid body allows to simulated a real rigid body subjected to physic's law. The gravity factor was set equal to zero, which means that the ball is not subject to gravity, also the friction has been set to zero, otherwise the ball would have been pruned and stopped after having covered a certain space, and the rotation with respect to the z axis is blocked.

- The script as previously mentioned has a very important role for the logic of the game. When the ball is instantiated, the start function is called, in which the ball is tied with the high game elements, which are the advisory zones and the socket. This operation is necessary to allow the scripts to share variables within the editor.

  - the "goball" method throws the ball to the right or to the left at the chosen speed which will remain constant throughout the game.

  - The OnCollisionEnter2D function is a function integrated in Unity that is triggered every time a collision occurs, depending on the object hit the ball calls a different function, and sends the information to the socket.

  - The ball can instantiate two types of explosions, a positive one,

shown in Figure 43, that represents the end of the round without errors while the second negative (Figure 44) that is instantiate when an error is made by one of the two players (or the back wall was hit, or it was wrong colour in the sequence)

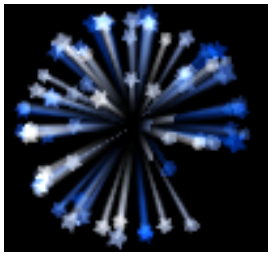

Figure 43: Fireworks explosion, in case of positive end

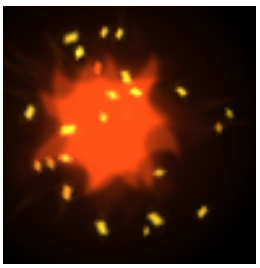

Figure 44: Red explosion, in case of negative end

– The hitfactor function influences the bounce of the ball depending on how it hits the racket (Figure 45), following the rules of the original pong; that are: if the racket hits the ball at the top corner, then it should bounce off towards our top border. If the racket hits the ball at the center, then it should bounce off towards the right, and not up or down at all. If the racket hits the ball at the bottom corner, then it should bounce off towards the bottom border.

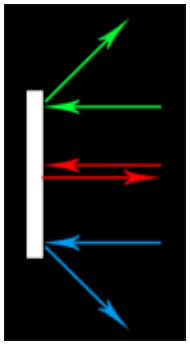

Figure 45: Hit factor on the racket

– The ball must also change state when a player calls the hint, the sprite is shown in Figure 46. The ball is stopped to allow the robot to have enough time to suggest to the player.Since the ball remains immobile, it is important that the player has visual feedback on the ball to inform him that his request for suggestion has been successful.

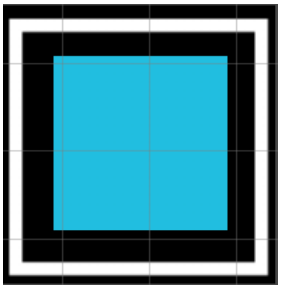

Figure 46: The ball stopped

The code, then when the suggestion request event is recorded, saves in a vector the direction and the speed of the ball. In this way, the ball can resume at the same speed and direction as when it was stopped.

5.6.5 Colours zone

The Colours zones, (Figures 47,48,49) are present only in the Place version, they must send the information to the Python code through the socket. To do this, they have been equipped with a triggered collider box, which means that when they come into contact with a racket of a player, the function OnTriggerEnter2D sends the message to the socket.

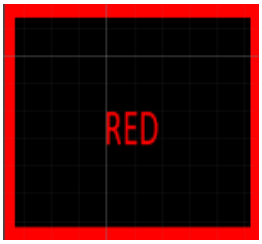

Figure 47: Red zone

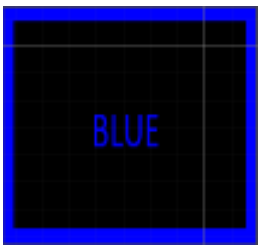

Figure 48: Blue zone

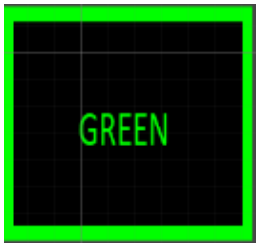

Figure 49: Green zone

5.6.6 Digital cube

The digital cube (Figure 50) was designed as an aid during the suggestion. In the tap version, its role is to illuminate the right colour when Cozmo hits it, thus simulating the behavior of real cubes in the players' possession.

Figure 50: Digital cube

6 Experimental evaluation

In the previous chapters it was discussed the design and implementation of the thesis. In this chapter it is explained the methodology used to test the game and the interpretation of the results obtained.

Questionnaires are one of the most common used and simplest tools for measuring the quality of a system.

Once the game has been implemented, particular attention has been taken in choosing the questions to ask the players since wrong questions lead to wrong results.

The questionnaire was then compiled keeping in mind that a game should be fun, and therefore some of the questions were included to verify the player's satisfaction. The remaining questions were developed by keeping in mind the objective of demonstrating the three hypotheses made before starting the design steps:

- the use of alternative gestures for interacting with game elements makes the users feel different levels of control over the game;
- the use of alternative gestures influences the user's perception of the game experience;
- the use of alternative gestures does not influence how the player perceives the presence of the robot and its role in the game.

The first hypothesis concerns the usability of the general system. To test this hypothesis the SUS questionnaire was used. The second and third

hypotheses concern respectively how the interaction with tangible objects a affects the player's perception and how the use of tangible objects affects the perception of the robot in the game system. To evaluate these hypotheses the GEQ (Game Experience Questionnaire) is used.

6.1 Questionnaire

The questionnaire was divided into five parts.

- A cognitive part where is evaluated the player's previous experiences regarding the use of both domestic and toy robots,their experience with augmented reality environments and their experience with video games;
- A part where the player can evaluate the performance of the system through the SUS (System Usabilty Scale);
- A part where the player is required to evaluate the use of tangible objects in the game and how the different interacting method in the different versions has changed their perception of the game system;
- A part that regards the perception that the players had of the robot in the game system and if their perception of the robot changes according to the version played;
- At the end, two open questions have been proposed, so that the player could express his opinion on the game more freely and suggest some features that could improve the gaming experience or possibly correct defects found in the system.

6.1.1 GEQ, Game Experience Questionnaire

The Game Experience Questionnaire has been developed by Eindhoven university of technology. It has a modular structure and consists of:

- the core questionnaire.
- the Social Presence Module.
- the Post-game module.

The GEQ was chosen to evaluate players' experiences with tangible objects and with the robot. The first part wants to measure flow, immersion, competence, tension and challenge components. Each of these components is measured for each version of the game, especially in the part of interaction with tangible objects. Following are some questions, adapted to the experience taken into consideration, presented by the questionnaire taken by the GEQ:

- I felt skilful;
- I enjoyed it;
- I found the game as difficult;
- I found the game challenging;
- I felt bored.

The second part evaluates the psychological aspects of the players' experience and the behaviours towards other entities, both virtual and real (such as the human opponent):

- I forgot everything around me;
- I felt frustrated;
- I felt completely absorbed;
- I felt challenged;
- I had to put a lot of effort into it.

The last part regards the post game, and the impressions that the players had immediately after having finished the game. Then, it evaluates the immersion of the players in the experience and their general impressions. Some examples of questions are listed below:

- My actions depended on the other(s) actions;
- I felt revengeful;
- I felt schadenfreude (malicious delight);
- I influenced the mood of the other(s);
- I felt jealous about the other(s).

6.1.2 SUS, System Usability Scale

The definition of "usability" is the quality of the user experience when using a product. To measure the usability it is necessary to adopt a standard method that acts as a unit of measurement for the usability of each tested system. It is important that the proposed questionnaire covers the following areas [16]:

- effectiveness (the ability of users to complete tasks using the system, and the quality of the output of those tasks);
- efficiency (the level of resource consumed in performing tasks);
- satisfaction (users' subjective reactions to using the system).

The SUS was created by John Brooke in 1986 through an experimental method, the questionnaire consists of 10 questions, the minimum necessary to cover the 3 areas mentioned above. Ideed a questionnaire too long could irritate or stress the tester, giving invalid results.The SUS is a likert scale of 10 questions. The questions are:

1. I think that I would like to use this gaming system frequently;
2. I found the gaming system unnecessarily complex;
3. I thought the gaming system was easy to use;
4. I think that I would need the support of a technical person to be able to use this gaming system;
5. I found the various functions in this gaming system were well integrated;
6. I thought there was too much inconsistency in this gaming system;
7. I would imagine that most people would learn to use this gaming system very quickly;
8. I found the gaming system very cumbersome to use;

9. I needed to learn a lot of things before I could get going with this gaming system.

The advantages of using the SUS system are many, in fact the system is very simple to administer and it gives valid result also using few testers.

6.1.3 Scoring SUS

To calculate the SUS score there is a precise mathematical method. The score goes from 0 to 100 and the average is considered the value of 68. Each questions' score contribution will range from 0 to 4. For items 1,3,5,7,and 9 the score contribution is the scale place minus 1. For question 2,4,6,8 and 10, the contribution is 5 minus the scale place. The sum of the scores needs to be multipled by 2.5 to obtain the overall value of SUS.

6.2 Design of the experiments

In addition to the care taken in drawing up the questionnaire, attention was also paid to the conduct of the experiments themselves.

It was decided to conduct the experiments in the following method.

- The players were divided into pairs, both players were asked to fill in the first part of the questionnaire, the one concerning the evaluation of their previous experiences.
- Both players were explained the rules of the game verbally, no demo was used.

- Players were invited to complete at least one game for each version, in addition the correct evaluation by the participants of the questionnaire both players had to ask for the help of the Robot at least once per game. Otherwise the game was repeated.

- At the end of each game the players were asked to change sides of playfield.

- Between each test the order in which the players played the different versions of the game was changed every time. It is logical to assume that the first version played has an lower average scores than the others, this is due to the fact that the players still have to get used to the game system.

6.3 Results

Results were analysed for statistical significance. To this aim, the Kolmogorov-Smirnov test was first applied to verify whether the distribution of the elements is a normal distribution and therefore subject to a hypothetical law. A Pearson correlation analysis was than carried out and the Pearson index P calculated. The Pearson index is always between 1 and -1:

- if $P > 0$ the variables are said to be directly related, or positively correlated;

- if $P = 0$ the variables are said uncorrelated

- if $P < 0$ the variables are said inversely related, or negatively correlated.

Twenty volunteers participated in the tests with an age between 20 and 29 years, (with an average of 23.35 years). All the volunteers were selected from university students.

The first part of the questionnaire was used to investigate the previous experiences of players regarding similar technologies tested in the game. The results obtained are:

- 63.2% said to play video-games regularly;
- 26.3% said to play video-games few times;
- 84.2% said to have never used these other toy robots;
- 63.2% said to have never used other kinds of service robots;
- 52.6% said to have used hand and body gesture-based control few times;
- 5.3% said to have used hand and body gesture-based control every day;
- 26.3% said to have never used hand and body gesture-based control.

The volunteers proved to be rather expert as regards the use of videogames, they are inexperienced regarding the use of toy robots, the experience improves regarding to the technologies of hand and body gesture-based control.

6.3.1 SUS results

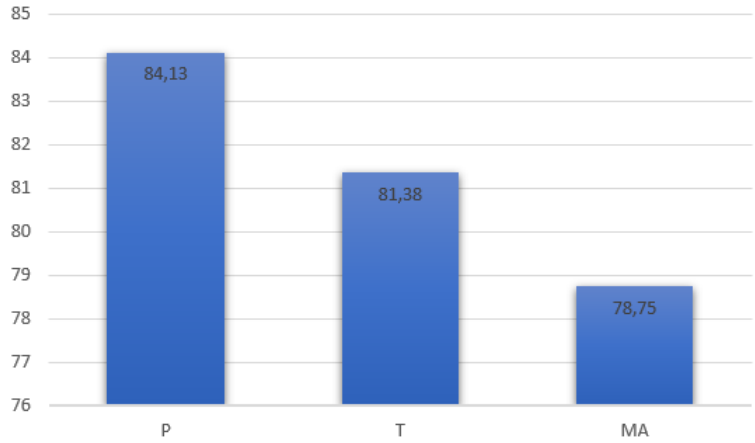

Figure 51: SUS results,variants are reported abbreviated as P (Place), T(Tap) and MA (Mid-Air)

The score given by the SUS (shown in Figure 51) is quite high for all three version. The best Place version scores are not surprising: in fact a greater usability was hypothesized since the Place version allows to change colour more quickly and there is no way of confusing the gesture to change colours with the gesture to ask the suggestion.

The disadvantage is that part of the gaming area must be dedicated to the control zones, thus limiting the game space for the players.

In the last there is the Mid-air version, penalized by the fact that some players had difficulty in executing the gesture in a correct way, especially in the first exchanges and in the most agitated phases of the game.

6.3.2 Tangible Interaction

In this section are analysed the results obtained form tangible interaction part.

The results, obtained regarding the opinion of the players if it was easy to confuse the action of changing colour with that of asking for the suggestion, are interesting (Figure 52).

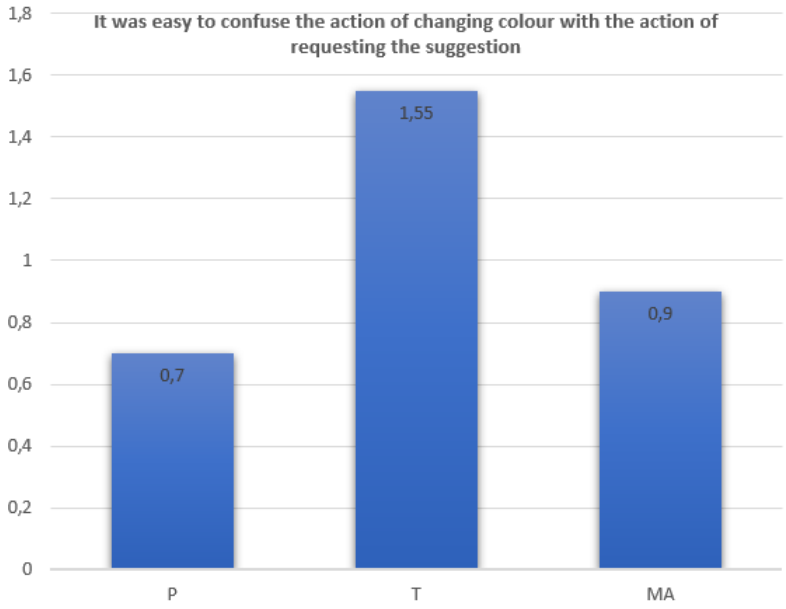

Figure 52: variants are reported abbreviated as P (Place), T(Tap) and MA (Mid-Air)

All the scores are satisfactory, but the results obtained indicate that in the case of multiple actions to be used with the same object it is not preferable to use the same repeated gesture (such as double tap) but replace it with a more natural action and possibly coherent with the action to be taken.

Results reported in Figure 53 show instead the ease with which players believe they can change colour, as evidenced by the SUS the score of the Place is the best.

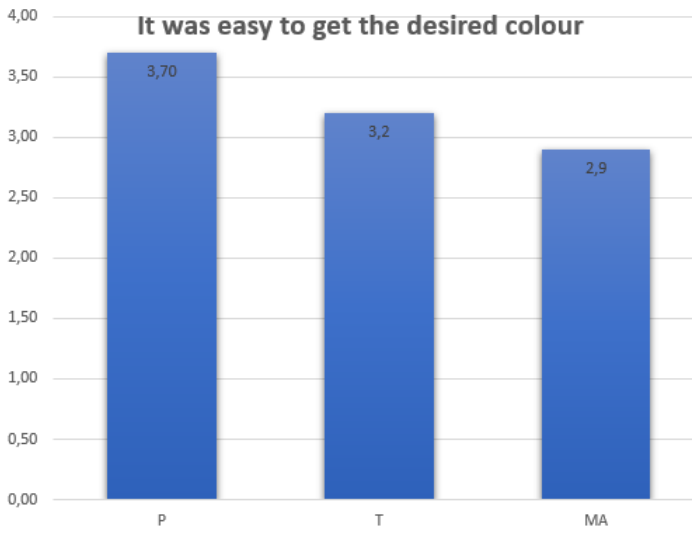

Figure 53: variants are reported abbreviated as P (Place), T(Tap) and MA (Mid-Air)

that are obtained similar results with the action of requesting suggestions to the robot (Figure 54).

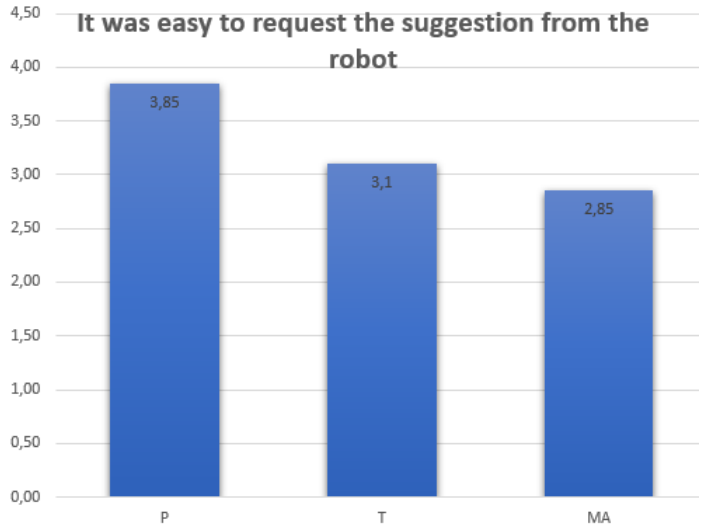

Figure 54: variants are reported abbreviated as P (Place), T(Tap) and MA (Mid-Air)

6.3.3 Robot Interaction

In this part of the questionnaire the interaction with the robot was evaluated. The results obtained are the same for all three versions, so it is possible to assume that the perception of the robot does not change depending on the version played. Anyway, the most significant results were as follows:

- $\mu = 3.55$ is the valuation about the perception of the robot as an intelligent part of the gaming system;
- $\mu = 2{,}9$ is the score about the central role of the robot in the game;
- $\mu = 2{,}55$ is the valuation about the lifelike being of the robot;
- $\mu = 2{,}8$ is the valuation about the personality of the robot;

- $\mu = 3{,}8$ is the score about the fact that they were able to hit the ball as they expected;
- $\mu = 3{,}7$ is the score regarding players' appreciation of the robot's movements during the game.

In all versions the robot has been indicated as an intelligent part of the system, this is a fundamental requirement of a robotic game and it has been satisfied, furthermore the robot has been evaluated as a central part of the system and therefore also the goal of making the robot a protagonist of the game has been reached. In conclusion it can be deduced that the robot was perceived as a fundamental part of the game in all three versions regardless of the gesture used to interact with it.

6.3.4 Game experience

About the game experience, the game was judged as satisfactory at stimulating the interest of players.

In all three versions the players found game challenging (μ=3,05) but not difficult (μ=1,5). This means that despite the simplicity of the rules, the game is able to provide a motivating challenge that kept the interest of the players high.

On the other hand,the game having been thought of as a competitive multiplayer, the game fails to bring out the competitive spirit of the players. In fact, to the question "What I did affected what the other(s) did" the response rate is $\mu = 2{,}4$ which indicates that the players did not pay much attention to the opponent. Also the question "I felt schadenfreude (malicious

delight)" which is the feeling that player feels when in a competition, the score is on average low: $\mu = 1.2$. However, the game was liked by the players in all three versions with a 3.5 rating.

Players were asked to give their opinion on which version they preferred most and the winner was the Place version:

- Place version: 55%;
- Tap version: 20%,
- Mid-Air version: 25%.

.

6.3.5 Open questions

At the end of the questionnaire, open questions were proposed to the players to highlight the aspects they liked in the gaming experience and those still to be improved. The questions asked were:

- Describe two things you liked the most about the game system/experience;
- Describe two things you would suggest to improve about the game system/experience.

From the answers to the first question it can be assumed that most of the players appreciated the role of the robot in the system, so the basic requirement regarding the centrality of the robot in the gaming system is confirmed.

It was also appreciative of the expressiveness of the robot and the gaming system, especially the fact of using physical objects to interact with the system instead of more traditional control systems like a joystick

Regarding the aspects to be improved the players suggested to increase the speed of the ball, for example by increasing it at each bounce, or to further expand the game system, i.e., give a way to recover lives or suggestions. Moreover they suggested to introduce a more direct physical interaction with it (e.g., "by giving high five for winning feedback").

The learned results show a solid and versatile gaming system suitable for developing new robotic games.

7 Conclusions

The designed game was evaluated in a positive way, so the requirements to implement a game that follows the PIRGs philosophy can be considered satisfied.

Players expressed appreciation for using tangible objects instead of a more traditional joystick, and the role of the robot in the gaming system, both fundamental requirements for the PIRGs paradigm.

Several experiments were carried out to verify the hypotheses conceived during the design of the game, later they will be repeated for greater clarity of exposition.

- the use of alternative gestures for interacting with game elements makes the users feel different levels of control over the game;
- the use of alternative gestures influences the user's perception of the game experience;
- the use of alternative gestures does not influence how the player perceives the presence of the robot and its role in the game.

The first hypothesis could be partially accepted, since SUS results showed substantial differences in the usability of the three different versions. Regarding the control of the game, it has been observed that the players have shown during the test phase only significant discrepancies between the action to change the colour of the cubes and the action to request the suggestion from the robot.

The second hypothesis was been rejected because the differences found in the various versions do not justify a different perception of the game, as the players have only highlighting the differences in the ease to obtained the desired colour and to ask the suggestion to the robot.

The third hypothesis could be totally accepted, indeed the robot's role being positively evaluated, as the tests clearly show that there is no correlation between the gesture and the robot perception.

Regarding to the gaming experience, experiments showed that players have shown a certain interest in the game system that is sufficiently engaging and challenging but it is necessary to make a note: the game was designed as a competitive player vs player. However the players' approach was more collaborative; there was no perceived competitiveness among the players, they seemed to be more interested in making the sequence of points as long as possible rather than beating the opponent. For the candidate's opinion, this fact can be attributed to the behaviour of the robot that highlighted the failures of the players with animations that showed sadness or anger, while the robot celebrated their successes with expressions of satisfaction or contentment. This behaviour may have induced the players to "satisfy" the robot instead of beating the opponent as is expected in the game system, leading players to collaborate rather than challenge each other.

7.1 Future work

The built game system turned out to be an interesting system (results highlighted by the SUS), so the game logic could be expanded following the suggestions given by the players.

Future work may be focused on increasing the number of testers both to highlight the positive results obtained and to confirm or definitively reject the second hypothesis, given that the differences obtained were not sufficient to confirm it.

It could be also considered the idea of increasing interaction both with the robot (as suggested by the players) but also with the virtual environment itself, for example by adding sound feedback for explosions or implementing an algorithm to recognize voice command.

8 Appendix

8.1 Questions

Players are asked to rate each question with a score ranging from 0 to 4 (0: strongly disagree, 4: strongly agree).

1. How often do you play video games?
2. How often do you use natural interface for hand and body gesture like Kinect, Leap Motion, etc?
3. How often do you use/interact with toy robots (like Cozmo, Sphero, etc.)?
4. How often do you use/interact with other kinds of social robots, such as home robots (like Roomba, etc.).
5. I think that I would like to use this gaming system frequently.
6. I found the gaming system unnecessarily complex.
7. I thought the gaming system was easy to use.
8. I think that I would need the support of a technical person to be able to use this gaming system.
9. I found the various functions in this gaming system were well integrated.
10. I thought there was too much inconsistency in this gaming system.

11. I would imagine that most people would learn to use this gaming system very quickly.

12. I found the gaming system very cumbersome to use.

13. I needed to learn a lot of things before I could get going with this gaming system.

14. I could get the colour I wanted.

15. It was easy to get the desired colour.

16. I always get the desired colour.

17. It was easy to request the suggestion from the robot.

18. It was easy to confuse the action of changing colour with the action of requesting the suggestion.

19. I felt frustrated by the interaction method.

20. The interaction method is pleasant to use.

21. The interaction method works the way I want it to work.

22. I was able to hit the ball as expected.

23. I was always able to determine the Place of the ball in the playing field.

24. I was able to move freely/the way i want in the respect of the playground.

25. I perceived the robot as an intelligent part of the gaming system.

26. I perceived the robot as an enemy.

27. In general, I felt the robot presence was crucial for the game.

28. I think the way the robot was acting distracted me from the main goal of the game.

29. I perceived the reactions of the robot as lifelike (i.e., not artificial).

30. I felt robot has a personality.

31. I felt like the robot was intentionally reacting to my actions.

32. I clearly understood the suggestion provided by the robot when needed.

33. I think the way the robot suggest me the colour was coherent with the game version.

34. I found the presence of the robot annoying.

35. I liked the way the robot was moving.

36. I clearly understood what the robot was saying.

37. I think that the fact the robot was talking was important for the game experience.

38. I wold have preferred a different way to provisioning the suggestion WITH the robot.

39. I wold have preferred a different way to provisioning the suggestion WITHOUT (instead of) the robot.

40. I would have preferred a more direct interaction with the robot (eg. Touching, hitting, ...).

41. I felt the robot and me were affectively linked (bonded) each other.

42. I felt disturbing that other players can ask help from the robot.

43. I felt challenged.

44. I found the game as difficult.

45. I enjoyed the game.

46. I felt skillful.

47. I felt bored.

48. I did not have a clear idea of how to achieve a given result.

49. The feedback was clear (lives, winning, etc.).

50. I felt completely absorbed (in the game experience).

51. I felt jealous about the other(s).

52. What the other(s) did affected what I did.

53. I felt revengeful.

54. The other(s) paid close attention to me.

55. I paid close attention to the other(s).

56. What I did affected what the other(s) did.

57. When the other(s) was(were) happy, I was happy.

58. I influenced the mood of the other(s).

59. I felt schadenfreude (malicious delight).

60. Overall I liked the game.

61. Overall, indicates which version of the game you preferred.

62. Describe 2 things you liked the most about the game system/experience.

63. Describe 2 things you would suggest to improve about the game system/experience.

References

[1] Python official site

`https://www.python.org/about/`

[2] zeroMQ official documentation

`http://zeromq.org/intro:read-the-manual`

[3] OpenCV official documentation

`https://docs.opencv.org/3.4/d1/dfb/intro.html`

[4] F. Gabriele Prattico', Alberto Cannavo', Junchao Chen and Fabrizio Lamberti. **User Perception of Robot's Role in Floor Projection-based Mixed-Reality Robotic Games** *Dipartimento di Automatica e Informatica, Politecnico di Torino* 2018

[5] Diana Pagliari and Livio Pinto. **Calibration of Kinect for Xbox One and Comparison between the Two Generations of Microsoft Sensors.** *National Center for Biotechnology Information,* 30 October 2015

[6] Nick Statt. **Humanizing smart robots for the masses.** *The Verge* 14 October 2016

[7] Lee Teschler **Cozmo Robot, a distant relative of Vector: Tear-down.** *Microcontroller Tips,* August 10, 2018

[8] OpenCv tutorials,Geometric Transformations of Images

`https://docs.opencv.org/3.0-beta/doc/py_tutorials/`

py_imgproc/py_geometric_transformations/py_geometric_transformations.html

[9] OpenCv tutorials,Changing Colorspaces https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html

[10] OpenCv tutorials,Morphological Transformations https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html

[11] Fabrizio Lamberti, Filippo G. Pratticò, Davide Calandra, Giovanni Piumatti,Federica Bazzano, Thiago R. K. Villani. **Robotic Gaming and User Interaction: Impact of Autonomous Behaviors and Emotional Features.** *Dipartimento di Automatica e Informatica, Politecnico di Torino,* Torino 2017

[12] Yoshihisa Nitta. **NtKinect: Kinect V2 C++ Programming with OpenCV on Windows10.** *Nitta laboratory, Tsuda college,* 19 July 2016

[13] Fabrizio Lamberti,Alberto Cannavò, Paolo Pirone. **Designing Interactive Robotic Games based on Mixed Reality Technology.** *IEEE International Conference on Consumer Electronics (ICCE),* Torino Italy 2019.

[14] D. Martinoia, D. Calandriello, and A. Bonarini **Physically interactive robogames: Definition and design guidelines,** *Robotics and Autonomous Systems, vol. 61, no. 8, pp. 739–748,* 2013.

[15] F. Gabriele Pratticò, Piero Baldo, Alberto Cannavò, and Fabrizio Lamberti. **Investigating Tangible User Interaction in Mixed-Reality Robotic Games.** *9th IEEE International Conference on Consumer Electronics pp. 1-6,* Berlin, 2019.

[16] John Brooke. **SUS - A quick and dirty usability scale.** *Usability evaluation in industry, pp. 189-194,* United Kingdom, 1996.

[17] W. IJsselsteijn, Y. De Kort, and K. Poels **The game experience questionnaire.** *Eindhoven: Technische Universiteit Eindhoven,* Eindhoven, 2007.

[18] Andrea Bonarini Francesco Amigoni Tiago Nascimento EwertonLopes. **Learning behaviors and user models to optimise the player's experience in robogames.** *Artificial Intelligence and Robotics Laboratory Department of Electronics, Information and Bioengineering of Politecnico di Milano,* 2018

[19] F. Gabriele Prattico, Alberto Cannavò, Junchao Chen and Fabrizio Lamberti. **User Perception of Robot's Role in FloorProjection-based Mixed-Reality Robotic Games.** *IEEE 23RD International Symposium on Consumer Technologies, pp. 1-6* , Berlin 2019

[20] Fabrizio Lamberti, Davide Calandra, Federica Bazzano,Filippo G. Prattico, Davide M. Destefanis. **RobotQuest: A Robotic Game based on ProjectedMixed Reality and Proximity Interaction.** *IEEE Games, Entertainment, Media Conference (GEM),* Torino, August 2018.

[21] Sophia Bernazzani. **What's the System Usability Scale (SUS) & How Can You Use It?** *Hunspot,* November 2018