ΤΕΧΝΟΛΟΓΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΣΧΟΛΗ ΚΑΛΩΝ ΚΑΙ ΕΦΑΡΜΟΣΜΕΝΩΝ ΤΕΧΝΩΝ

Μεταπτυχιακή διατριβή

PILOTTAR: A MULTI-USER AUGMENTED REALITY CARD GAME

Κωνσταντίνος Δεληγιάννης

Λεμεσός 2014

ΤΕΧΝΟΛΟΓΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΣΧΟΛΗ ΚΑΛΩΝ ΚΑΙ ΕΦΑΡΜΟΣΜΕΝΩΝ ΤΕΧΝΩΝ

ΤΜΗΜΑ ΠΟΛΥΜΕΣΩΝ ΚΑΙ ΓΡΑΦΙΚΩΝ ΤΕΧΝΩΝ

PILOTTAR: A MULTI-USER AUGMENTED REALITY CARD GAME

του

Κωνσταντίνου Δεληγιάννη

Λεμεσός 2014

ΕΝΤΥΠΟ ΕΓΚΡΙΣΗΣ

Μεταπτυχιακή διατριβή

PILOTTAR: A MULTI-USER AUGMENTED REALITY CARD GAME

Παρουσιάστηκε από

Κωνσταντίνο Δεληγιάννη

Επιβλέπων/ουσα καθηγητής/τρια: ...……………………………………………………………

Μέλος επιτροπής: ………………………………………………………………………………

Μέλος επιτροπής: ………………………………………………………………………………

Τεχνολογικό Πανεπιστήμιο Κύπρου

Δεκέμβριος, 2014

Πνευματικά δικαιώματα

In this section I would like to mention the people who helped me and guided me for the production of this project. I would like to express my gratitude to my supervisor, Dr. Charalambos Poullis, who offered me priceless support and assistance throughout all the stages of my project. Also, an honourable mention goes to all my colleagues and friends who have helped me on testing my application, recognising errors whilst providing me ideas for implementation.

ABSTRACT

Nowadays, the use of Augmented Reality technologies is rapidly growing. Using Augmented Reality we can inject virtual data and objects into the real physical environment enhancing collaboration and interaction. Moreover, mobile devices were also advanced with high computing and graphical processors. This enhancement allowed AR technology that needs high computing calculations to be expanded on mobile devices increasing its portability factor.

This research thesis investigates Augmented Reality technology applied in mobile games. Furthermore, an AR card game was developed to examine the realistic aspect of the technology. Also, this AR game supports multiple users investigating as well the networking inside this technology. The aim of this project was to discover whether such an AR game can replace the traditional game. The procedure used for developing the game is explained in detail.

After developing the game, an evaluation was conducted. Investigation results are outlined and the findings are discussed. Finally, future development expansion and corrections are described.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY - ABBREVIATIONS

2D:	Two-Dimensional
3D:	Three-Dimensional
API:	Application Programming Interface
APK:	Android Package Kit
AR:	Augmented Reality
GPS:	Global Positioning System
GUI:	Graphical User Interface
HMD:	Head Mounted Display
HUD:	Heads-Up Display
PC:	Personal Computer
PDA:	Personal Digital Assistant
RPC:	Remote Procedure Call
SDK:	Software Development Kit
USB:	Universal Serial Bus
VR:	Virtual Reality

1 Introduction

Technology made vast and rapid development steps during the past 50 years. Technological achievements are everywhere in the world and we cannot imagine our lives without them. Terms such as Telephony, Internet and Virtual Reality have been fully developed to support our work, enhance our gameplay experience and generally to facilitate our everyday life. Nevertheless, the progression of technology is not stopping here and is always making improvements on existing technologies or introducing new things to expand their potentiality.

An innovative technology that made its entrance into our life are mobile devices and smartphones. Smartphones have become the extension of our hand allowing us to make various actions quickly and wherever we are. These actions vary from, communicating with other people, to taking photos and surfing on the internet. Moreover these devices are equipped with high-end hardware that a decade before could be found in big desktop PCs showing this rapid development of technology. With such a powerful hardware, smartphones can support complex operations and expensive graphics for gaming.

Furthermore, Augmented Reality is one of these new areas that promise new ways for people to view the world. The primary goal of this technology is to "inject" virtuality into reality allowing us to observe and interact with computer generated data and objects created with "harmony" into the real environment. The term was firstly introduced by Professor Tom Caudell in 1990s that developed a head mounted digital display that could help workers assemble electrical wires in aircrafts (Lee, 2012). Combining these two new developed technologies mentioned above, Augmented Reality can become portable and accessible to everyone.

Because AR was evolved through VR, is often linked with HMDs but nowadays this is not the case because as stated above mobile devices can support such a "heavy" functionality. Researchers are exploring new ways to provide this technology to everyone. An interesting area that AR and smartphones can be merged is gaming. Augmented Reality can take mobile gaming to a higher level with games that combine realistic and immersive virtual information while allowing players to move around in their real surrounding instead of focusing only on the screen. However, most of the games created for AR incorporate characters and objects that are unrealistic and sometimes do not match with the real environment as shown in Figure 1.1 where a virtual ocean is augmented on the table. By creating an AR game that can augment, as realistic as possible, virtual objects while the players can easily interact with them without any hesitation, could possibly replace many existing games.

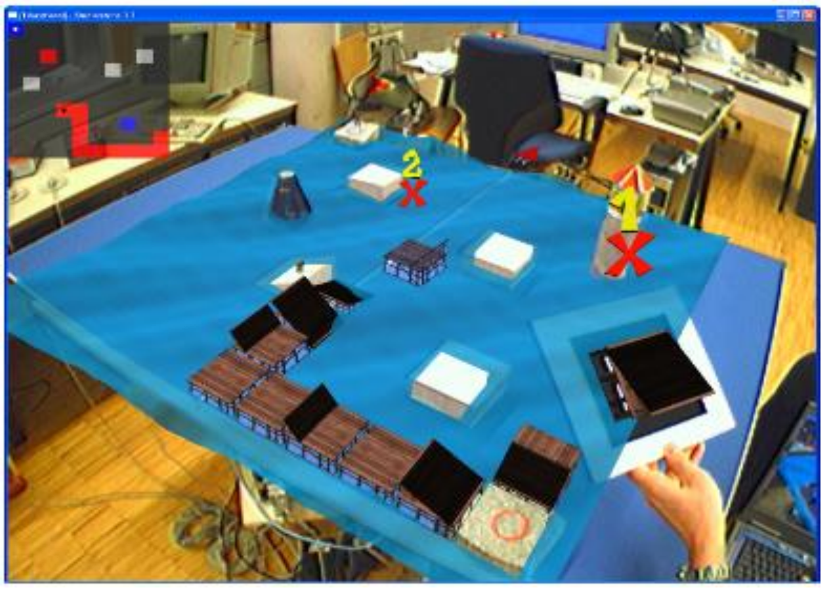


Figure 1.1: MonkeyBridge (Barakonyi, Weilguny, Psik, & Schmalstieg, 2005), augments an ocean where users build a virtual bridge to cross to the other side.

1.1 Focus of this work

This union of AR and mobile devices started developing in the recent years so many AR applications and games came out in mobile app stores (Google Play, Apple store). Although, most applications are dealing with navigation or providing information for guidance and games that do not employ most of the AR technology capabilities (Tan, & Soh, 2010).

Moreover, mobile AR technology is divided into two main categories: marker-based recognition and pose computation (Uchiyama, & Marchand, 2012). In the pose computation approach, the only requirement needed is the camera’s orientation and viewpoint. By taking this information, the virtual objects are augmented in front of the camera image and depending of the usage they can be aligned on the real objects. Marker-based approach contrarily, renders the virtual data when a predefined marker is tracked and recognised. This marker can vary from a real object to an image.

This project will be focused on the marker-based approach which is characterised to be practical under specific limiting circumstances in order that new and more feasible ways to make this approach more portable can be explored.

Furthermore, as mentioned before, existing AR games do not provide realistic virtual objects and characters so the aim of this study is to create an AR game, based on a real physical game, that will render realistic elements to "trick" the user that is playing the real version of that game. That led to the thought of creating a card game, since the objects that are used in the actual game and must be augmented in the application, can easily be closer to physical objects.

The goal of this study is to find out how augmented reality can be applied in such a multiuser application (game) in order to be as immersive and realistic as the actual game. Moreover, this thesis aims to assess the behaviour of the players of such games and their preferences in the game. Analysing these, will help in the creation of an augmented reality game that will try to incorporate as many of the identified positive factors and avoid or try to address any negative ones.

More specifically, the aims and objectives are listed below:

> Can Augmented Reality, implemented in gaming, fully replace existing materialistic games?
>
> This involves the following subtasks:

- In depth study of AR's strengths and limitations.
- Test and evaluate hardware that can be used for this technology
- Test and evaluate software and APIs that can be used for developing such a game
- Design and develop an AR game using the appropriate components from above
- Evaluate the game with real users
- Review evaluation for further improvement
- Evaluate the final version of the game based on the Research question

1.3 Thesis Structure

The chapters of this Thesis report are:

Chapter 2 – Literature Review: In this chapter, previous work and research on Augmented Reality and mobile gaming is examined so that an understanding of both subjects will be gained and provide guidance for developing the desired game.

Chapter 3 – Requirements Management: This chapter provides the initial requirements' specification needed for designing the game and an overview of the system giving a better understanding on how the game is played and what events must be included.

Chapter 4 – Design and Implementation: On this chapter, user interface design is analysed and an initial prototype is presented. Additionally the software tools that are used for development are discussed. When the basic structure of the final design is created, the steps of creating the final game are analysed in detail.

Chapter 5 – Evaluation: The chapter describes how the evaluation of the game was conducted and discusses the results.

Chapter 6 – Conclusion: Final chapter presents the conclusions of this project and discusses the possible improvements and future work.

2 Literature Review

2.1 Introduction

In order to obtain a full perception of all the aspects of my project, a background research on previous and current software and technologies on the field need to be conducted. This research is divided into three parts. The first part examines various aspects of general AR technology. Part two analyses existing games that use AR technology. The third part discusses some specific AR games and applications that are more suitable for the current research question and also what software technologies as well as hardware are required for creating the project's application.

2.2 Augmented Reality Technology

AR is a fast growing research technology in the field of Virtual Reality. The term refers to the annotation of virtual information (objects, text etc.) inside the real environment through the appropriate hardware where humans can perceive mostly through the sense of sight (Feiner, 2002; Thomas, 2012).

Silva, Oliveira, & Giraldi (2003) discuss various characteristics of AR systems and essential devices for such technology. As mentioned in the article, AR is defined by three properties; a) Mixture of real and virtual environment. b) Real – time interactivity. c) Registration in 3D (Mackay, 1998; Azuma, Baillot, Behringer, Feiner, Julier, & MacIntyre, 2001). Through the rapid development of technology, another property that is also highly considered is portability. Head – Mounted – Displays were the first devices that could use AR technology (Chuptys, & De Coninck) but in terms of portability HMDs were ineffective because many AR applications require users to be more active while playing. That was the reason that led developers in designing and creating portable and inexpensive HMDs such as the Oculus Rift (Desai, Desai, Ajmera, & Mehta, 2014) and google's cardboard (Google Cardboard, 2014). Nevertheless, most of the new developed HMDs belong to the Virtual Reality technology so for evolving them into AR some extra features need to be added such as a camera to display the real world on the HMD's screen.

Except, from HMDs Augmented Reality was developed on handheld devices as well. Daniel Wagner Dieter Schmalstieg of Vienna University of Technology (Wagner, & Schmalstieg, 2003) describes the first stand-alone AR system with self-tracking running on an unmodified personal digital assistant (PDA) with a commercial camera. The application was developed with a well-known marker based toolkit (ARToolkit) so it accurately registers overlays in the scene. Therefore, AR technology was expanded into smartphones which are more powerful these days providing a bigger screen as well (Henrysson, & Ollila, 2003).

By taking seriously the portability factor and the expansion of the technology into smartphones, AR was then spread into new areas. An area where AR is considered to be a very powerful weapon is Marketing and Advertising. AR marketing joins together the real world with the virtual, computer-generated objects enhancing what can be seen by users resulting in the enrichment of the product (Bulearca, & Tamarjan, 2010). Such enrichment can be seen in an AR application created by a shoe company as shown in Figure 2.1 which allows the user to see 3D virtual photorealistic object of the real shoe. This provides a memorable content for the consumer which is often called experiential marketing.

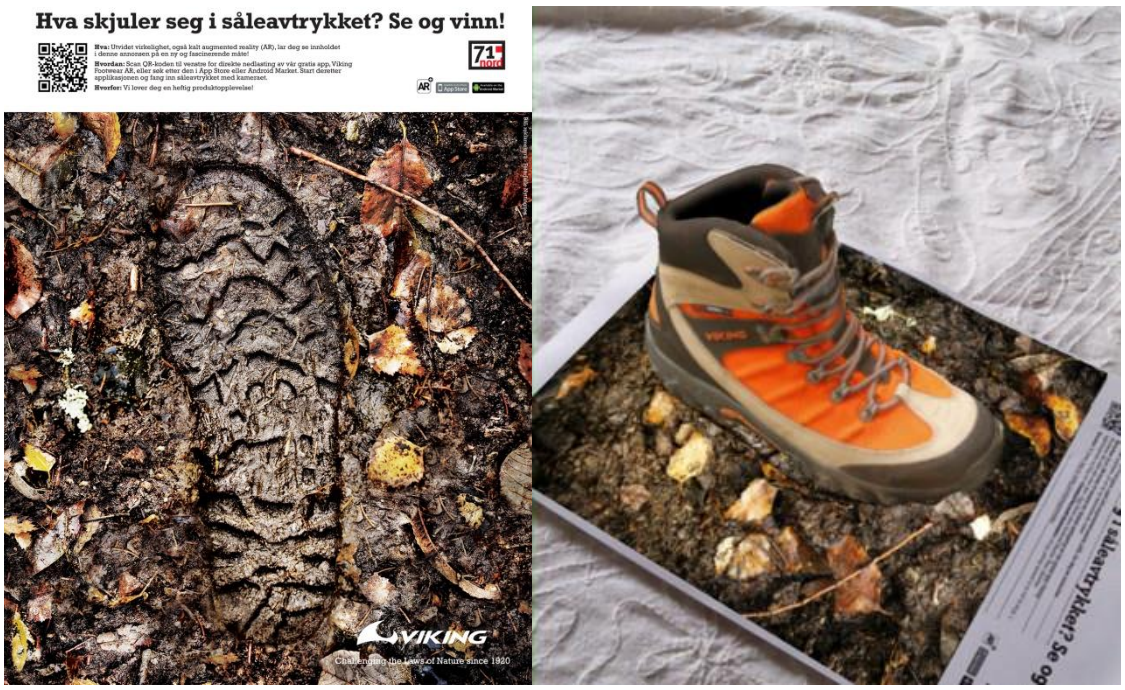

Hva skjuler seg i såleavtrykket? Se og vinn!
Hva: Utvidet virkelighet, også kalt augmented reality (AR), lar deg se innholdet i denne annonsen på en ny og fascinerende måte!
Hvordan: Scan QR-koden til venstre for direkte nedlasting av vår gratis app, Viking Footwear AR, eller søk etter den i App Store eller Android Market. Start deretter applikasjonen og fang inn såleavtrykket med kameraet.
Hvorfor: Vi lover deg en heftig produktopplevelse!
71° nord
AR
App Store
Android Market
VIKING
Challenging the Laws of Nature since 1920

Figure 2.1: Viking Footwear AR, readers of the magazine download the application and scan the image to see the virtual model of that boot while they can explore additional shoes of that company.

Välkkynen, Boyer & Urhemaa (2011) presented an Augmented Reality system where the users could be informed about a product's information using their mobile phones. Also, except from the additional information (web-based content, videos) that is displayed about that product, a 3D model is constructed in order to give the consumer the exact product without all that packaging (Figure 2.2).

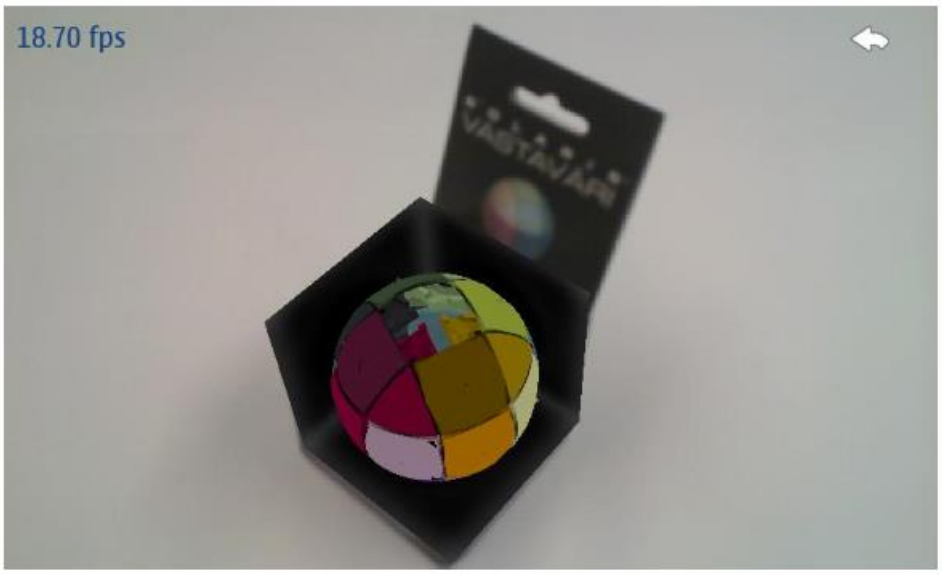


Figure 2.2: A virtual 3D object is augmented on top of the package to allow users see how the inside product looks like (Välkkynen et al., 2011)

The system that was created applies the vision-based augmented reality which uses unique markers on each product allowing the 3D model to be placed exactly on top of the physical product. This is more preferable from position-based AR because it gives a more immersive experience to the consumer. The first main functionality of the system is allowing the consumer to “peek inside the box”, by constructing a 3D model of the product and some virtual parts of the packaging to make it realistic. In that way the consumer can see the product from different sides by simply rotating the product. The next important functionality is displaying to the consumer various information about the product such as a video related to the product and anything that is available in an external database. Lastly, the system contains a simple location-based context-awareness (CA) module which discovers where the application is being used. If it is used at home, it displays different information about the product such as its user manual; otherwise if it is used at the store it displays the product’s specification.

Another important subject that AR has a high impact on is education and training. Lee (2012) summarises different applications and software using AR technology on these two subjects. AR is a powerful tool in subjects such as astronomy, chemistry and biology providing information that are inaccessible or very expensive without it; for example an AR body anatomy lesson provides real scale 3D computer-generated models of body organs.

2.3 Augmented Reality applied in mobile gaming

As stated above, AR can be applied on a wide variety of subjects and for this research it will be further examined in the field of Gaming. The game industry is characterised as the most suitable implementation of AR technology (Tan, & Soh, 2010). By using the real world for any kind of gameplay gaming is expanded in a new dimension. Nilsen, Linton & Looser (2004) examine the pros and cons of the game styles between real, computer and AR worlds, by comparing them in four aspects; physical, mental, social and emotional. Mixed Fantasy is a term used to describe how users experience gameplay by showing a proportion of real, virtual and imaginative content (Stapleton, Hughes, & Moshell, 2002). More precisely in real, physical games players mostly use real and imaginative content where computer world games use virtual and imaginative only. Here is where AR makes its entrance by combining all these three features to complete the user's experience.

Starting with the physical aspect of gaming, computer games are narrow in terms of physical interaction which is only focused on the hardware used and not with the whole body like real games. AR games will let players get more active allowing them to play a game in any real space, and not sitting in front of a screen, by inserting virtual objects into the real world.

Next, the mental aspect allows users to use their reasoning and thoughts to solve various complex situations. Computer games, because of the hardware, can support very complicated scenarios with a large amount of data, providing interesting scenarios and intelligent agents; allies or enemies. Physical games do not have such complicated scenarios because users must analyse a situation by themselves but can provide spatial reasoning by examining an object from all perspectives (3D space). AR can merge these two scopes allowing users to decide and to reason more precisely.

Thirdly, the social aspect in computer games is provided through networking allowing games to have a large amount of players communicating with each other and also because of the anonymity, users can create virtual characters of themselves. Socialising in real games allows users to communicate face to face. Although it cannot provide vast number of players like in computer games, it allows the players to communicate with other cues like gestures and gaze that can be very important in many games. As previously, AR can provide both communication methods but the biggest challenge is how to combine them into one application where all players can cooperate equally; both players communicating through networking and players communicating face to face (Billinghurst, Belcher, Gupta, & Kiyokawa, 2003).

Regarding the last aspect, the emotional aspect, in computer games it is provided through the graphics and audio that can trigger various feelings that will remain even after finishing the game (sympathising a virtual character). In real games, these emotions can be succeeded with all the senses and by controlling real objects in the environment (lights, speakers). Stapleton et al. (2002) recommends the use of real and virtual content, in order to have a fully emotional range game.

As technology advances, handheld devices such as tablets and smartphones are becoming more and more powerful, with fast computational and high-end graphic processor units making AR technology more accessible to mass audience (Figure 2.3) (Chang, Koh, & Duh, 2011). Wagner & Schmalstieg (2007) describe what technological constraints and user interface design aspects need to be considered when designing and developing an AR game on handheld devices.

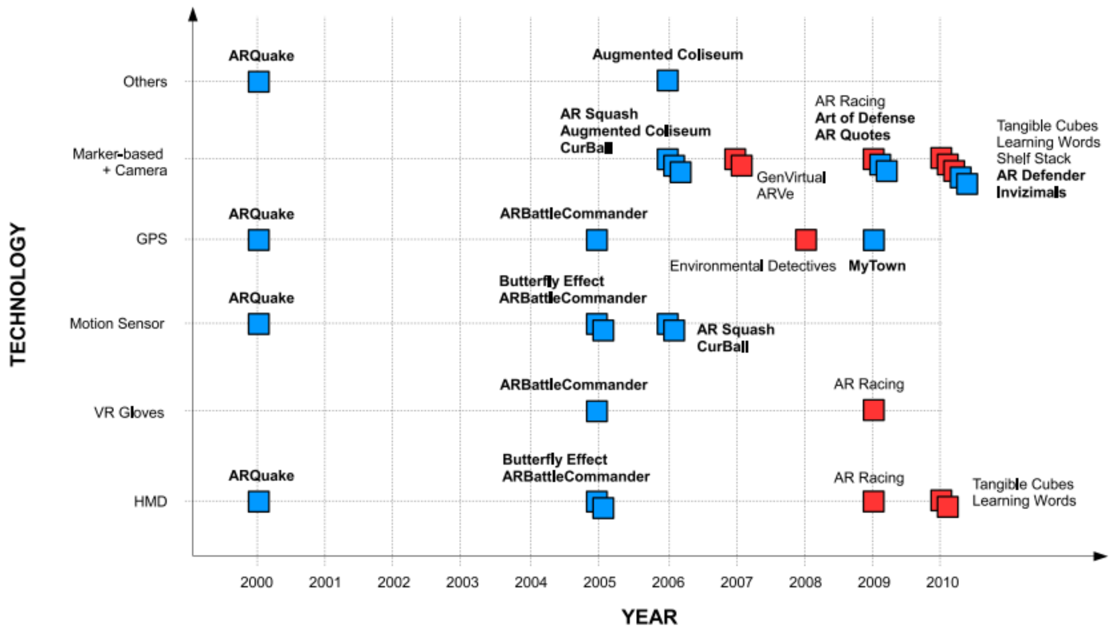


Figure 2.3: The graph is a review of various existing Augmented Reality (Tan, & Soh, 2010) (Blue is for entertainment and Red for serious games) showing the evolution from expensive hardware (HMD) to simple marker-based with camera devices.

Starting with the technological constraints, handheld devices do not have that power yet to support high quality graphics like in consoles and PCs but can still be competitive providing other kind of playing experience (mobility, camera, GPS) (Rubino, Michaluk, Nickinson, & Ritchie, 2014; Lavender, & Gromala, 2012). Wagner and Schmalstieg (2007) state that handheld devices are limited in tracking the user's pose because only a few devices are equipped with the appropriate sensors but nowadays that is not the case because almost every mobile device is equipped with such sensors like GPS, accelerometer and gyroscope. Considering the usability factor, handheld devices are much more portable and handier than a HMD but still tablets can be characterized uncomfortable for an extended period of time. Nevertheless, touch screen functionality can expand AR technology into new areas because users can actually control virtual objects with no use of extra hardware.

Moving on user interface design, the first thing that needs to be considered is the device that is going to be designed for. Because of the small field of view of handheld devices, users are observing, at the same time, content from the real environment around the screen which lowers the immersion level. This leads designers to consider other approaches of how such devices can be used in augmenting the environment. The first approach is mostly used with marker-based applications where the screen represents a new virtual environment and users can interact with virtual objects while not being very active with the device (Figure 2.4). The second approach is by making the device an interaction medium between real and augmented world often called "magic lens" (Baricevic, Lee, Turk, Hollerer, & Bowman, 2012). This allows users to be more active by moving the device around changing the viewpoint on the virtual world while the virtual content is still aligned together with the real one (Figure 2.5).

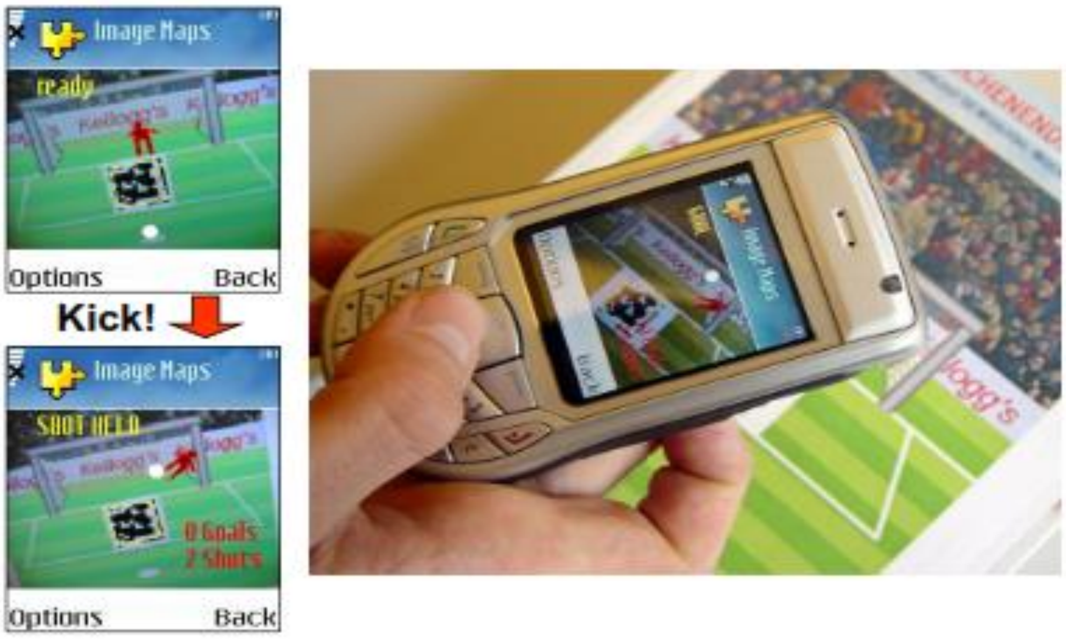


Figure 2.4: "Penalty Kick" (Rohs, 2007), the screen shows a virtual goalkeeper inside the field image which is used as a marker and the user aims by rotating and tilting the phone.

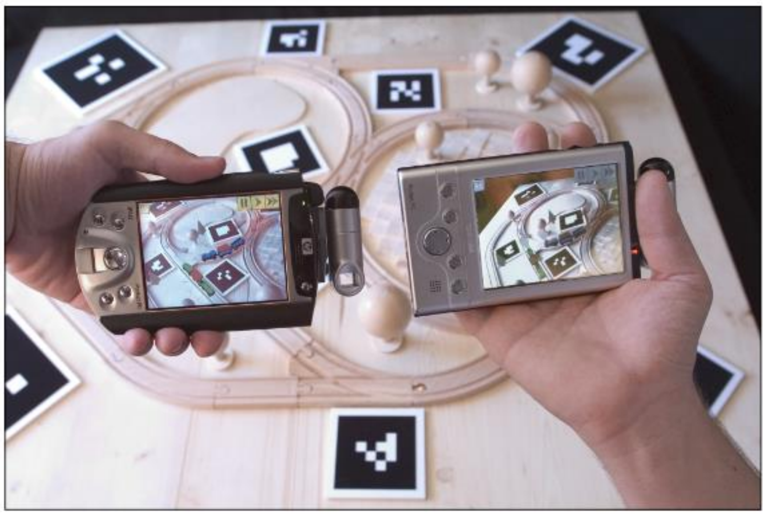


Figure 2.5: "The Invisible Train" (Wagner, Pintaric, Ledermann, & Schmalstieg, 2005), the user can interact with specific virtual trains and rail switches only when points the device at the railway and markers around it.

Supporting multiple users is another important factor that needs to be considered while designing an AR game. Multiplayer functionality in an AR application can be divided into two categories

The first category is by sharing a device with multiple users (Wagner, & Schmalstieg, 2007). A turn by turn based game can be very uncomfortable for games that require more than two players. The only advantage is that only one device needs to be used and configured.

The second category is by using networking, Bluetooth or functionality. This is further divided into co-located users and separated users. Szalavári, Eckstein & Gervautz (1998) designed a system where multiple co-located users could play the Chinese game Mah-Jongg. The game is played by four players and each player draws and discards tiles (cards) until they complete a legal hand. The system supports both social communication, where the players can talk with each other, and the freedom of individual player's privacy. To support the social communication each player wears a see-through HMD so except from verbal communication, any gestures from other players will be visible as well. Additionally, a Personal Interaction Panel (Szalavári, & Gervautz, 1997) consisting of a pen enables users to interact with the virtual objects and at the same time maintain the private information (tiles) of each user (Figure 2.6). The shared information for all players is succeeded with client-server architecture. The server keeps the game's entire graphical database and runs the simulation based on the input data. Each client (user) sees the corresponding graphics (personal tiles, tiles on the board, dice) and each manipulation that he does, it is broadcasted over the local network in order for all the players to be updated. The results, after testing the system with real users, showed that, although the hardware setup is not suitable for non-lab controlled testing, users found it very fun to play and even inexperienced users learnt quickly how to play the game with no extra information.

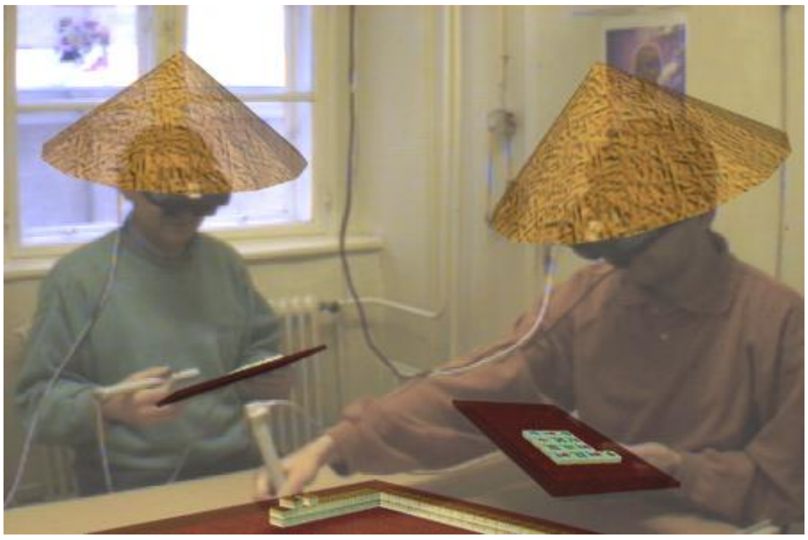


Figure 2.6: A view of the Mah-Jongg game from a player's view (Szalavári, Eckstein & Gervautz, 1998)

Next, Second Surface (Kasahara, Heun, Lee, & Ishii, 2012), is an Augmented Reality system that falls into the category of separated multi user application. Users can interact in real-time with contents generated by other users over physical surroundings. These generated contents vary from 3D drawing to texts and photos and intend to allow users to share their expression without vandalising a physical environment or object such as trees.

Firstly, the system uses image-based AR recognition technology to recognise a natural image from an AR dictionary set (trackable objects) which is shared among the users. Then, based on an estimation of the target's object pose, the pose of the device is defined and virtual canvas where the generated content is placed and finally rendered based on the target's object pose. All the devices are connected through a server which manages the data that need to be stored and also to be pushed on other devices that are facing the same object. Such a system would be helpful for explaining the use of many physical objects (trashcans) without adding extra physical items but instead adding virtual objects.

2.4 Augmented Reality games review

To start with, a multi-user AR card game was developed by researchers in Virtual Environments and Robotics Lab of the Monterrey Institute of Technology of Mexico (Diaz, Alencastre-Miranda, Munoz-Gomez, & Rudomin, 2006). Their aim was to develop a networked card game where people around the world could enjoy playing, incorporating real video with the virtual environment and the use of extra hardware component to interact. "Memory" game was chosen as it is a very popular and very simple to implement game. More precisely each user is equipped with a real board with cards which is connected on the computer. When a player flips a real card the sensors understand the action and a 3D model appears on the virtual board which is displayed on players' screen (Figure 2.7). For each pair that is found the players earn a point and when the game is finished the virtual 3D models are randomly replaced on the cards.

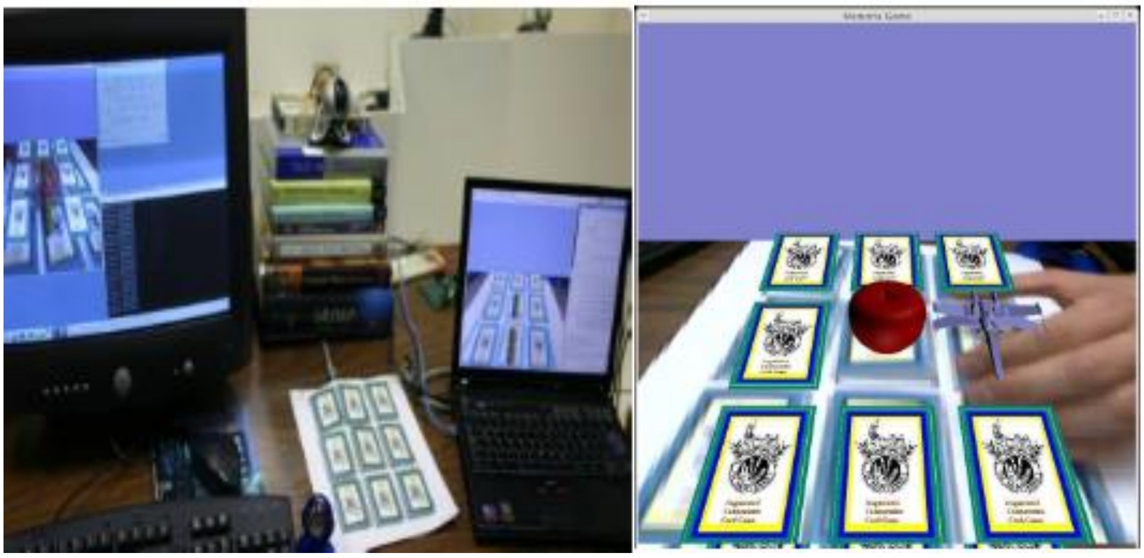


Figure 2.7: Memory AR (Diaz et al., 2006), game setup with the real board of cards (left), the augmented environment of the board (right)

Following, in the School of Interactive Computing and GVU Center of Georgia Institute of Technology a group of researchers developed two AR co-located games for handheld devices; Bragfish (Xu, Gandy, Deen, Schrank, Spreen, Gorbsky, & MacIntyre, 2008) and Art of Defense (Huynh, Raveendran, Xu, Spreen, & MacIntyre, 2009). Starting from “Bragsfish”, researchers aimed to develop an AR game with which they could investigate how social and physical interaction into a virtual world is linked to the use of a handheld device. The system consists of a big board which is used as the marker and the players aim with their camera handheld devices on that board trying to navigate their boat around and catch fish with the use of the device’s buttons. Additionally, players can steal fish from other boats increasing their skills. All players are aware of sharing a physical space, around the board, and a virtual world, “inside” the board, increasing this way the social interaction between the players (Figure 2.8).

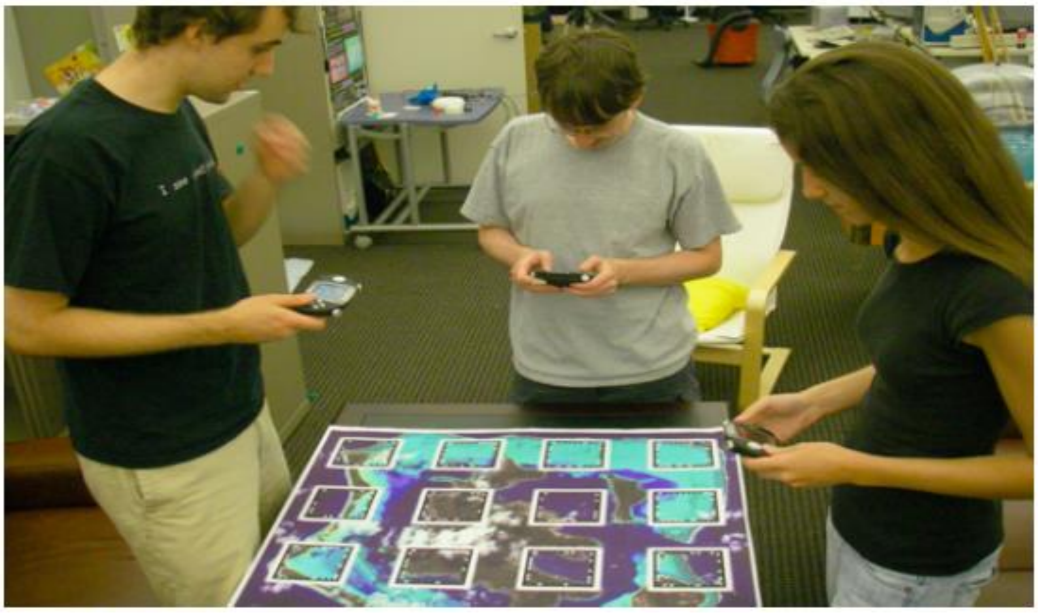


Figure 2.8: BragFish (Xu et al., 2008), game setup with three players

The physical interaction is achieved through visual, aural and physical cues from the shared environment (Figure 2.9). For example, when a fish bites a player's bait vibrations trigger other players' attention to decide what moves to make, either go steal the fish or keep waiting for their baits. The evaluation showed that most users preferred to play the game with real users and not computer ones and also with people that they know (friends and family). The issues that were shown was that by moving the device quickly around the board the tracking was lost for some seconds and also by standing and holding the device on top of the board was uncomfortable after some time.

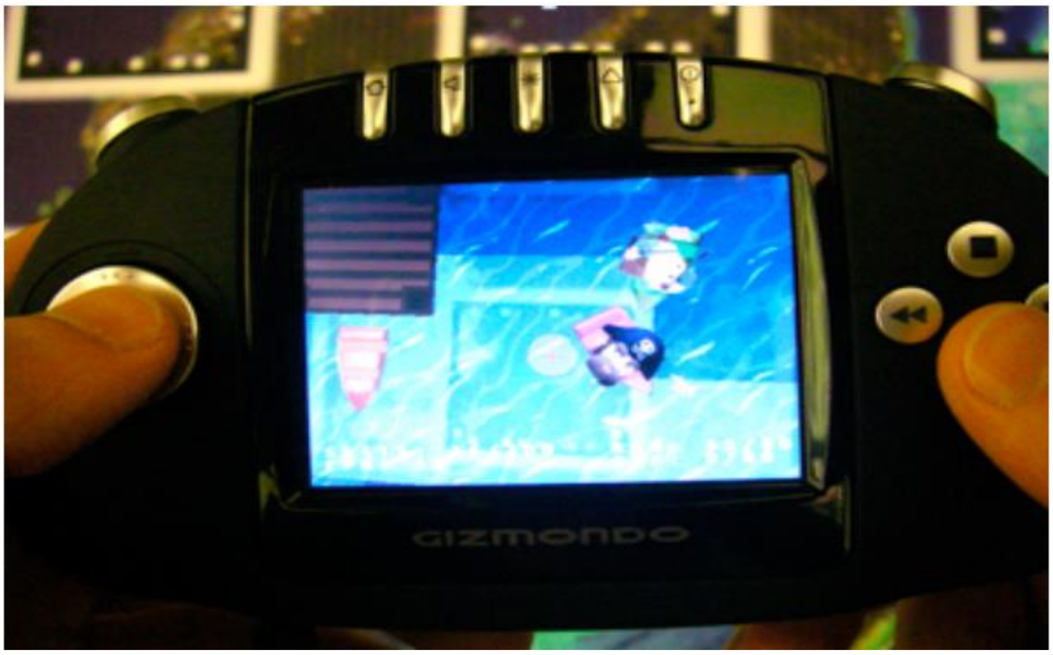


Figure 2.9: BragFish (Xu et al., 2008), augmented view from player's handheld device

Moving on the next application, Art of Defense is an AR tabletop board game for handheld devices where physical game pieces are used to create a physical and virtual world at the same time. Their research aims to find what are the advantages and restrictions of handheld devices on collaborative social AR games. AOD is a strategy based game where two players cooperate to defend their ground from enemies' attacks. The game takes advantage of the limited field of view of mobile devices because strategy games use "fog-of-war" to prevent viewing in unoccupied areas. This turns a possible restriction into an advantage. The next problem that is discussed is the lack of balance between the portability of mobile devices and the fixed marker-based game board. To overcome this restriction, they created a number of small hexagon shaped marker tiles that when put all together they create a map with the defending tower in the middle and for exploring new areas of the map, players have to change the positions of the tile. With this design architecture a dynamic and yet portable marker-based board is created. After testing the game with real players, results showed that the game was fun to play and the cooperating part was very interesting. Nevertheless some issues were addressed, such as the loss of tracking the marker and the lack of information about the enemy because of the limited view of the map. An interesting issue mentioned about the game was that sometimes a player blocked the view of someone else which in BragFish was considered as part of the game were players intentionally blocked the opponent's view to have a better position themselves. Hence, this issue can be considered when designing a game to be included as a part of it, expanding the advantages of a handheld device.

Okada & Arakawa (2010) from the Kyoto Sangyo University aimed to develop an augmented reality user interface which will guide beginners in learning the rules of a specific card game, without the help of experienced players, and evaluate the effectiveness of the system. The authors chose two card games: Mate and Bohemian Schneider. The system configuration is consisted by a pc and its monitor, a board with AR markers (cards) and a webcam which scans the card markers. The way the system works for the first game (Mate) is by guiding the beginner on which card suit he has to open depending on what the opponent had opened. Similarly, for the second game (Bohemian Schneider) the system recognises which card was thrown by the opponent and guides the beginner on which card he has to throw in order to win the hand.

Both games were evaluated by beginner players who completed a questionnaire after playing the games initially using the system for some rounds and then without it. The results have shown that the system did partly help the players to learn the rules and play the games but it also requires more experiments and more research for more functional augmentation methods.

The approach of Molla & Lepetit (2010) was creating a setup where a real – scene (a board) would be augmented on a computer screen generating a different game. Using Computer Vision techniques, the board and the pawns are located and enriched with virtual elements on the screen (Figure 2.10).

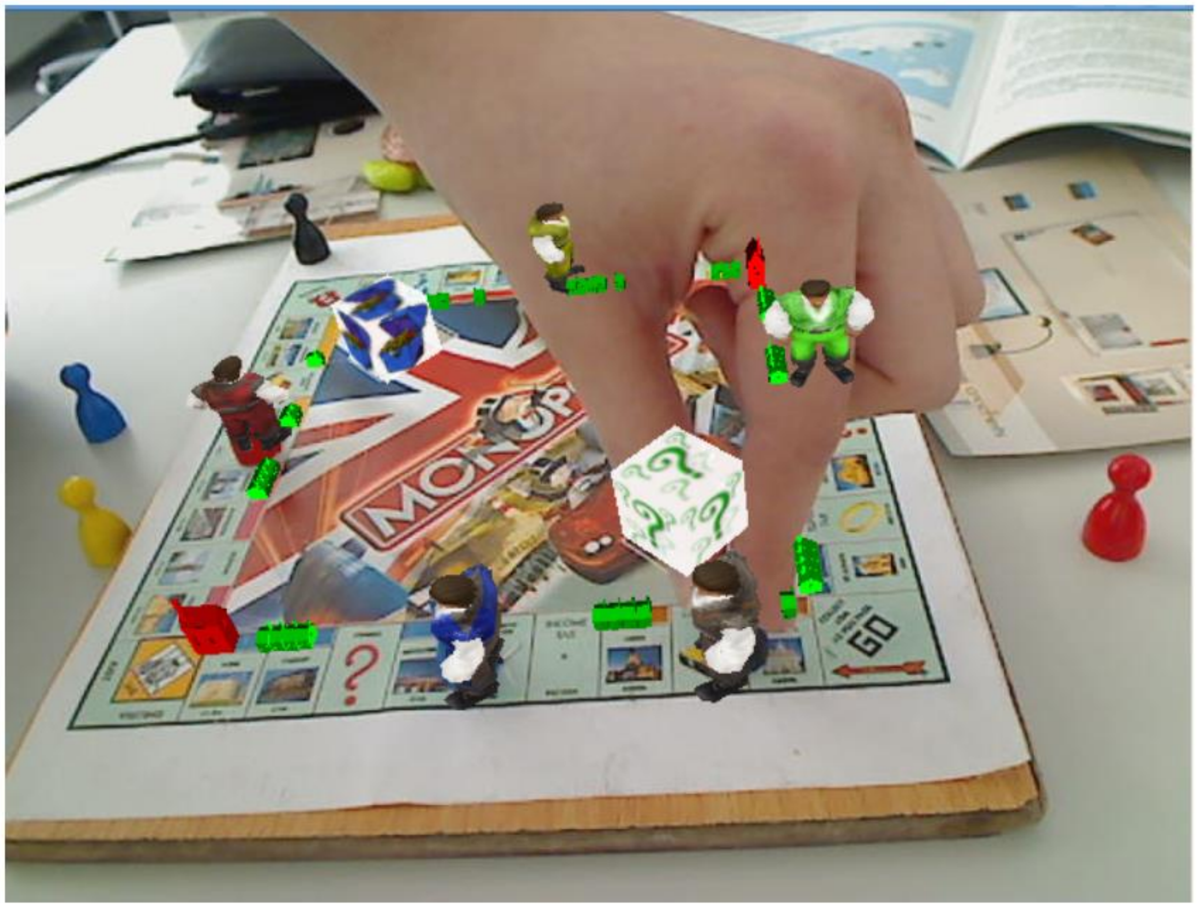


Figure 2.10: MonopolyAR (Molla, & Lepetit, 2010), the augmented view of the board showing the virtual pawns

The board detection is achieved by matching some featured points of a reference image of the board with some of the captured image. This will also restrict the detection of pawns only on those that are located on the board. To locate the pawns on the board a large number of 3D bounding boxes is generated in valid places on the board and “Viola and Jones” object detector finds pawns’ location. The colour is derived using a voting scheme where the pixels of the bottom part of the bounding box vote for a colour (yellow, red, green, blue and black) based on its hue and intensity. The colour with the most votes is selected. The research concludes by mentioning that AR technology can enrich already existing board games and make them more attractable and enjoyable.

2.5 Conclusion

All appropriate articles and related work was reviewed and analysed as shown above. Firstly, the general AR technology was discussed followed by the methods on how this AR technology can be applied on mobile gaming. In the last section specific AR games were reviewed on how these games applied these methods. Despite the fact that successful AR applications have already been reported in the literature, these primarily involve basic AR functionality and interaction i.e. simple annotation of information based on GPS location or basic instruction on augmenting virtual objects. Furthermore, networking is limited in AR applications and more precisely in games that nowadays this functionality plays an important role in the success of the game. After, reviewing the literature review, the requirements were exported as discussed in the following chapter.

3 Requirement Management

3.1 Introduction – System Overview

This chapter presents the systems requirements. Figure 3.1 shows the methodology that was followed and all the steps conducted in order for the game to be built which are discussed on the current and following chapters. Moreover, this chapter analyses all the event handling for the game as well as the specific rules handling that the chosen card game must follow.

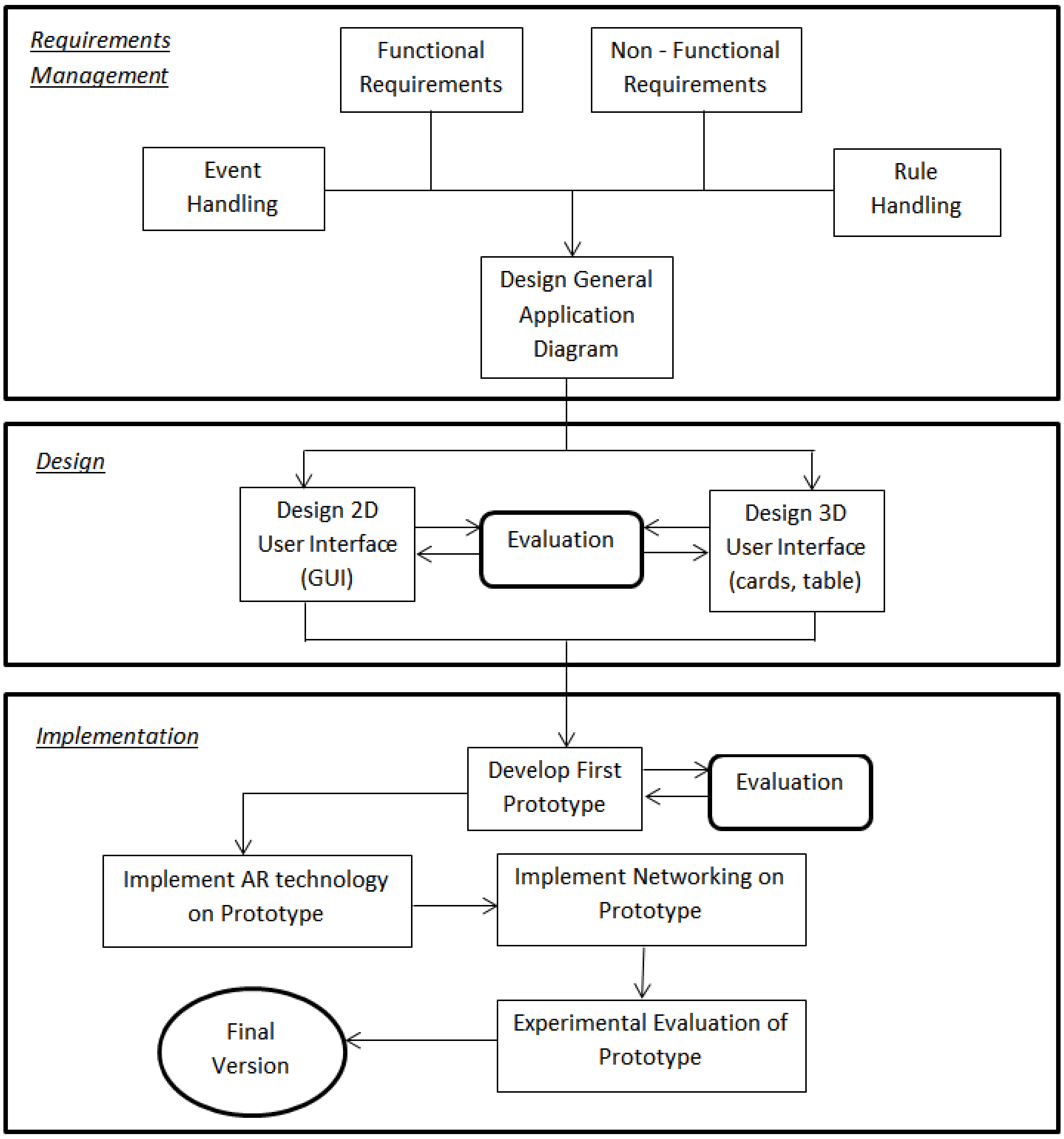

Requirements Management
Functional Requirements
Non - Functional Requirements
Event Handling
Rule Handling
Design General Application Diagram
Design
Design 2D User Interface (GUI)
Evaluation
Design 3D User Interface (cards, table)
Implementation
Develop First Prototype
Evaluation
Implement AR technology on Prototype
Implement Networking on Prototype
Final Version
Experimental Evaluation of Prototype

Figure 3.1: Methodology Diagram, all the steps followed for developing PilottAR

3.2 Requirement Analysis

Firstly, an investigation (ANNEX A1) was conducted to understand users' behaviour while playing card games and what are their preferences about the game. This was helpful in making the application as realistic as possible. Based on the results of this investigation (ANNEX A2) as well as the research on the related work, a list of requirements was exported and divided into two categories; Functional where are all the requirement features needed in order for the game to satisfy the user and operate correctly and Non-Functional where are all the requirements that are needed for the application to be expandable, adaptable and user friendly.

The method used for this part of the project is the MoSCoW (Vestola) technique which helps prioritise the requirements in an essential way. All the requirements are defined based on research and investigation into related work while reviewing various users' needs. Prioritizing the requirements helped to implement the features in the right way based on the importance of each requirement. Defining requirements is a really important step and needed to be done in an early stage to achieve the best possible output.

M: Must have this. (Top priority)

S: Should have this if at all possible. (Highly desirable)

C: Could have this if it does not affect anything else. (If time allows)

W: Would like to have this. (Not today)

3.2.1 Functional Requirements / Preliminary

<u>Network</u>

Table 3.1: Network Requirements

No.	Requirement	Priority
	Online Game	MUST
	Connect / Join the game	MUST
	Share cards played through the network	MUST
	Share cards declared through the network	MUST
	Share score through the network	SHOULD
	Share bidding through the network	COULD
	Chatting integration	WOULD
	Social media integration	WOULD

<u>Augmented Reality</u>

Table 3.2: Augmented Reality Requirements

No.	Requirement	Priority
9.	Allow gameplay only when AR marker is tracked	MUST

10.	Rich in detail AR marker (quick tracking)	MUST
11.	Use vary of images as possible markers (database of markers)	COULD
12.	Use of AR virtual buttons	WOULD

Rules

Table 3.3: Rules Requirements

No.	Requirement	Priority
13.	Throw same card suit as the first played	MUST
14.	Throw higher rank trump card if available	MUST
15.	Throw trump card if you do not have a card suit as the one first played	MUST
16.	Automatic count of points	MUST
17.	Change cards ranking power when trump cards suit is declared	MUST
18.	Automatic count of declarations	COULD
19.	Allow to show declaration only if available	WOULD

Gameplay

Table 3.4: Gameplay Requirements

No.	Requirement	Priority
20.	Turn-by-turn gameplay (anti-clockwise)	MUST
21.	Touch interaction with cards	MUST
22.	Only one player deals and shuffles	MUST
23.	Start round only when dealer declares the game (suit, points, team)	MUST
24.	Wrong gameplay handling (Error handling)	SHOULD
25.	Realistic throw of cards	SHOULD
26.	Realistic physics on collisions	COULD
27.	Declare names of teams	WOULD

3.2.2 Non-Functional Requirements

<u>User Interface</u>

Table 3.5: User Interface Requirements

No.	Requirement	Priority
1.	Simple understandable GUI	MUST
2.	Quick responsive gameplay	MUST
3.	Readable text	MUST
4.	Realistic textures	MUST
5.	Convenient positioning 3D elements	SHOULD
6.	Lag free	SHOULD
7.	Attractive GUI	COULD
8.	Appropriate icons on buttons	WOULD

3.3 Event Handling

In this section, all the events that are required for running the game correctly are described (Figure 3.2). First of all, all four players are gathered around the table, where the marker is in the middle, and decide which player is going to handle the virtual writing and shuffling (dealer) and which will be the maximum score. Then, each one opens the application on their device. The dealer creates the game and the other three players join that game. When all the players are connected, the dealer shuffles and deals the cards to the players and the bidding round starts.

In the bidding round each player bids or passes, depending on their cards, in order to have the game on their team side. The bidding is done in the same way as the traditional way of playing which is by talking on a turn-by-turn way. When three passes occur then the dealer declares the game and the main game begins.

The main game, as well as the bidding round, is a turn-by-turn game where the player who started bidding throws a card followed by the other three players. When all four players played their hand, the player who threw the highest ranking card on that round, collects the cards and plays first on the next round. This procedure continues until all the cards have been thrown and collected.

When the main game finishes, the cards collected by each team are counted and a total is summed up. Then the dealer adds on that total any extra points if they were declared on the main game. Depending on the total score of each team, the winning team of that round is decided and the appropriate score is "written" for each team. If any of the two teams have reached the maximum score which was predefined then that will be the winning team of the game else the game restarts from the bidding round.

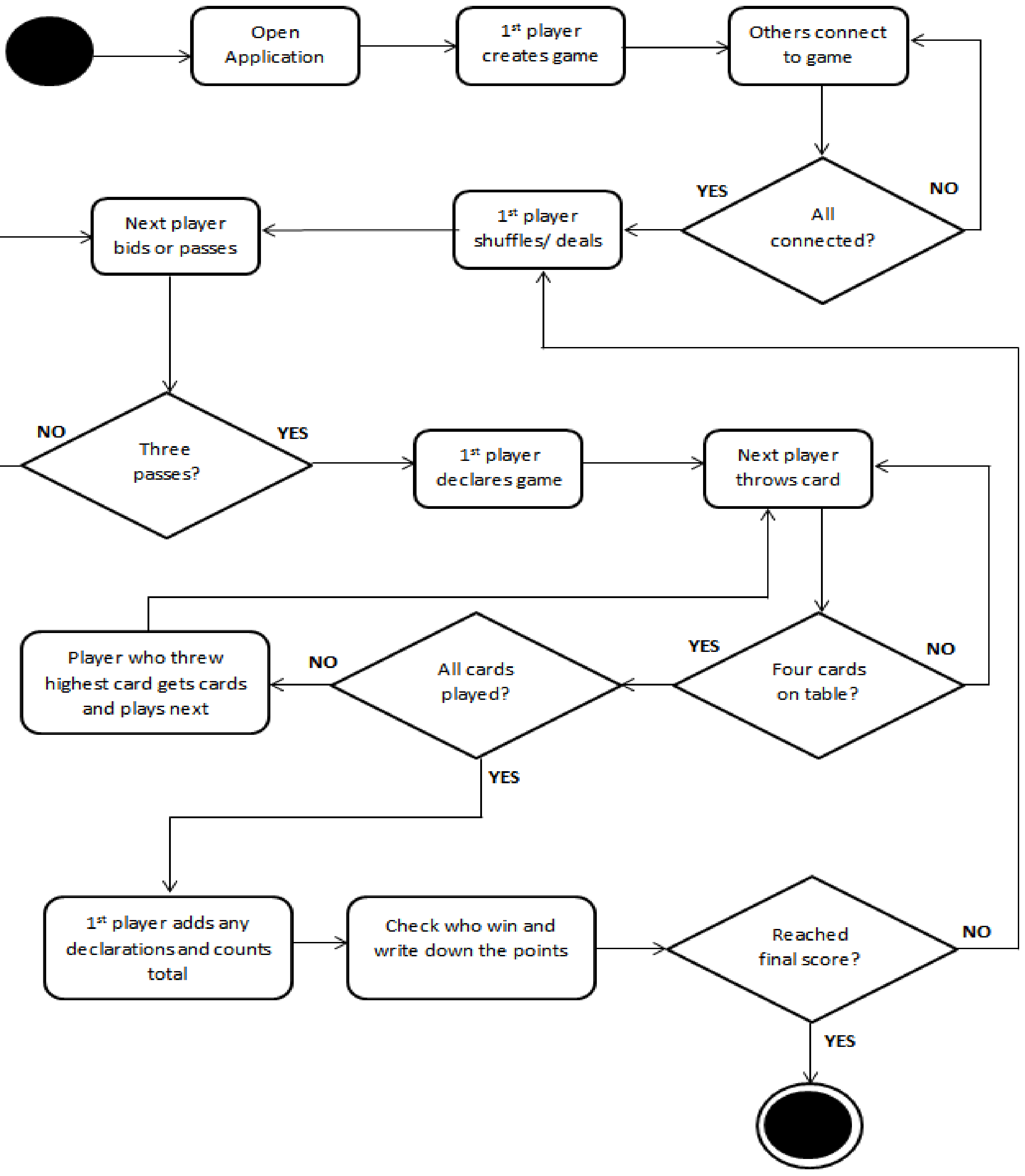


Figure 3.2: Overall gameplay events diagram

3.4 Rules Handling

In the traditional game of Pilotta there are some specific rules that must be followed in order to be played correctly. These rules must be included in the AR game as well as some other rules and restrictions which will handle any possible unintentional wrong playing from the players and will allow the correct flow of the game. The rule handling is divided into the three steps of playing which is the bidding, the main gameplay and counting the points on each round.

Starting with the bidding round, when the dealer comes to declare the bid of that round some restrictions must be implemented so that the main game can start correctly (Figure 3.3). First of all, when declaring the final bid, the bid must be between the numbers eight, which is the starting bid, and eighty, which is the maximum number that any team can bid with their cards. Secondly, the dealer must declare which card suit is the final bid (trump cards) and also which team bid last. These two restrictions are not included in the traditional game but for such a game, implemented on mobile devices, must be included so that specific calculations on the main gameplay can be computed. Additionally, an option for "closed" or "double-closed" must be available if any of the teams declares it (definition on ANNEX A4). When all these requirements are met the main game can start.

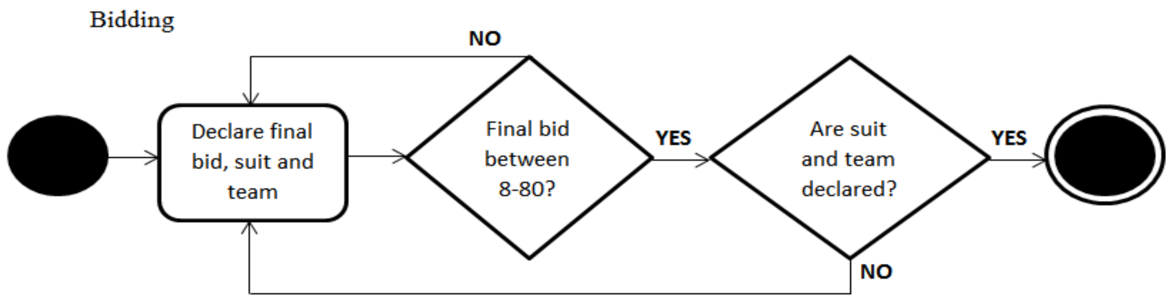


Figure 3.3: Bidding rules diagram

In the main gameplay, the rules are mostly about what card the players have available to throw (Figure 3.4). In the traditional game, the players know, outset, which card to throw but sometimes they throw a wrong card which leads on restarting the round which can be very annoying for the players. This error can and must be handled by the AR game with some specific calculations based on the game rules in order to allow the players choose only the correct cards. The 1st player can throw any card and depending on the card that the player throws, all the other players must throw the same suit card. When it comes to the next player there are some restrictions on which card that player can choose to throw. Three possible scenarios can occur. The first scenario occurs when the first card thrown is a trump card. If it is a trump card then a check determines if there are any trump cards on the player's cards. If there are no available trump cards then the player can throw any other card of their choice but when the player holds trump cards further checks must be calculated. These checks must determine if any of the player's trump cards is higher ranking from the cards on the table. Then the player must throw one of their higher cards (overtrumping) or else throw one of their lower cards (undertrumping). Continuing, the second scenario occurs when the first card is not a trump card. This means that the player must throw a card with the same suit as the first card played. A check determines whether the player, holds cards of that suit and allows the player to throw only one of them. The last scenario occurs when the results of the check on the second scenario are negative. Then the same checks as the first scenario are calculated. This process in the traditional game is called trumping.

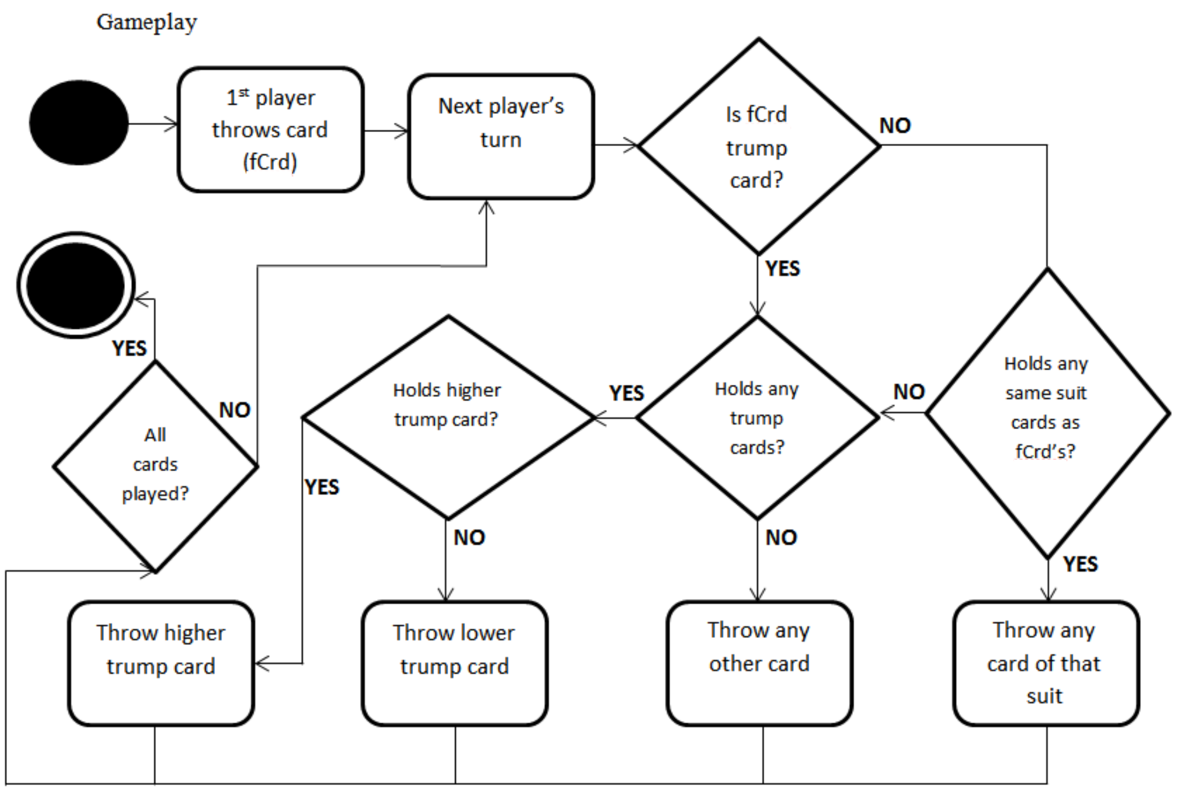


Figure 3.4: Main gameplay rules diagram

Another rule that must be followed in the main game, besides of throwing the correct card, is to add any available extra points on the teams. The first rule is to add ten extra points on the team that wins the last hand. A check determines which player threw the highest card on the last round and adds the points to the respective team. The other rule adds twenty points to the team which has, what is called, Belote. Belote is called when any of the four players have on their cards the King and the Queen card of the trump cards. A check determines if any of the players have Belote in their cards and adds the extra points to the respective team.

When the main game finishes, the total points for each team must be calculated depending on the cards that each team has collected (Figure 3.5). Then the dealer adds any declarations that occurred on the main game to the total of each team. In order to decide the winner of the round, a check determines if the total points of the team that bid last are equal or higher from their bid and depending on the result the score for each team must be "written down".

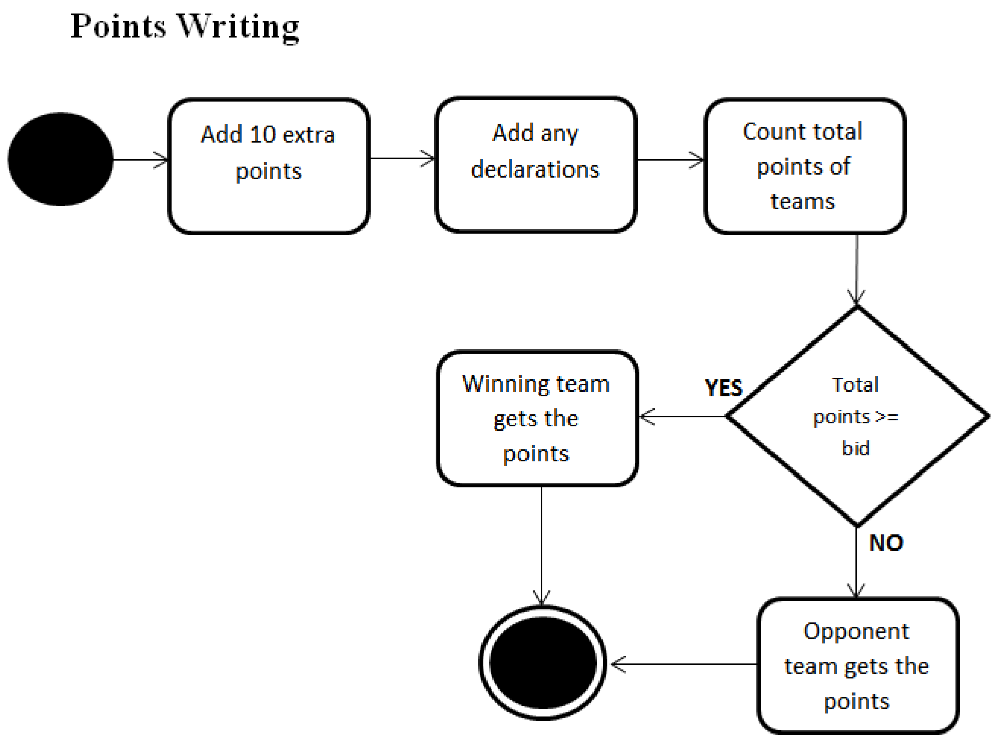

Points Writing
Add 10 extra points
Add any declarations
Count total points of teams
Total points >= bid
YES
Winning team gets the points
NO
Opponent team gets the points

Figure 3.5: Points writing rules diagram

3.5 Conclusion

In this chapter, all the requirements were listed based on the results of the first investigation and the literature review. Moreover all the events of the game as well as the rules that are used were discussed. Based on these requirements, event handling and rules handling the developing of the project's game was initialised.

4 Design and Implementation

4.1 Introduction

Upon completion of the requirement analysis, the next step is to design the initial idea as well as to develop the final application. The procedure that was followed is discussed and analysed in depth in this chapter. This section introduces all the technologies used and all the procedures followed for creating an AR game. Moreover, all the theoretical knowledge described in the previous sections is put in action, merging all the technologies required under one application.

4.2 Software and Tools Used

In this Section, all the software that was used to implement the source code for the application and design the user interface is going to be described. Also, any other tool that was used to create or edit various components essential for the application will be analysed.

4.2.1 Unity3D

The main software that was used for developing "PilottAR" is Unity3D game engine. It is a developing tool not only for games but for almost any application (architectural visualizations, animations etc). The Unity game engine is developed by Unity Technologies in Denmark and integrates a custom rendering engine with the nVidia PhysX physics engine and Mono, the open source implementation of Microsoft's .NET libraries (Craighead, Burke, & Murphy, 2007). As shown in Figure 4.1 Unity3D is the leading engine used in the global game engine market.

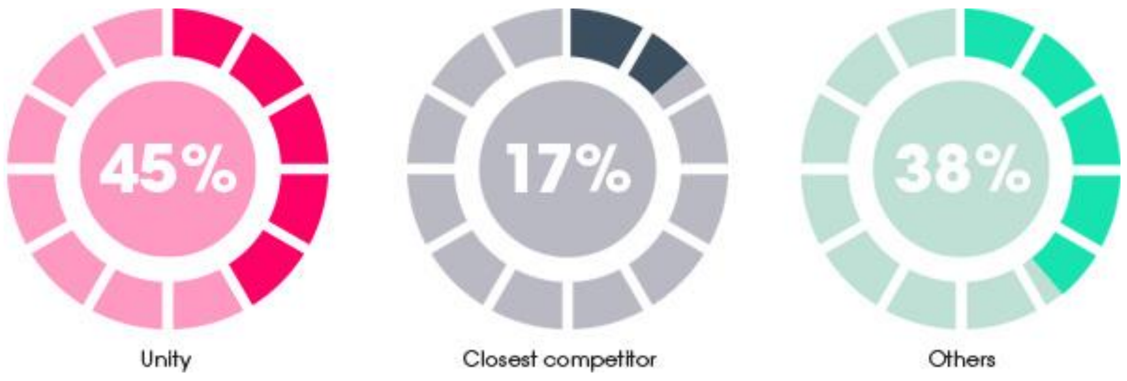


Figure 4.1: Global game engine market share (THE LEADING GLOBAL GAME INDUSTRY SOFTWARE, 2014)

Moreover, it was chosen because it offers many useful and powerful features. First of all, is a cross-platform engine allowing developers to export their application in a variety of platforms such as Windows, Android, IOS, and Xbox without any, or with minor, changes in their codes. Secondly, it offers high documentation with many examples for all its functionalities. This is a very important aspect because for every difficulty that occurred in respect of coding or any other feature, it ensures that there is a possible solution for overcoming them. Furthermore, it supports multiple programming languages including C#, JavaScript and Boo. This allows a programmer to choose a programming language that is more confident with, while mixing them as well. Next, Unity’s drag-n-drop functionality in its editor empowers developers for very fast designing the general environment of the application. It is a very user-friendly feature because users know and see exactly where everything is, without defining it dynamically, while assigning various components (scripts, material, physics) by just dragging them on the object (Figure 4.2).

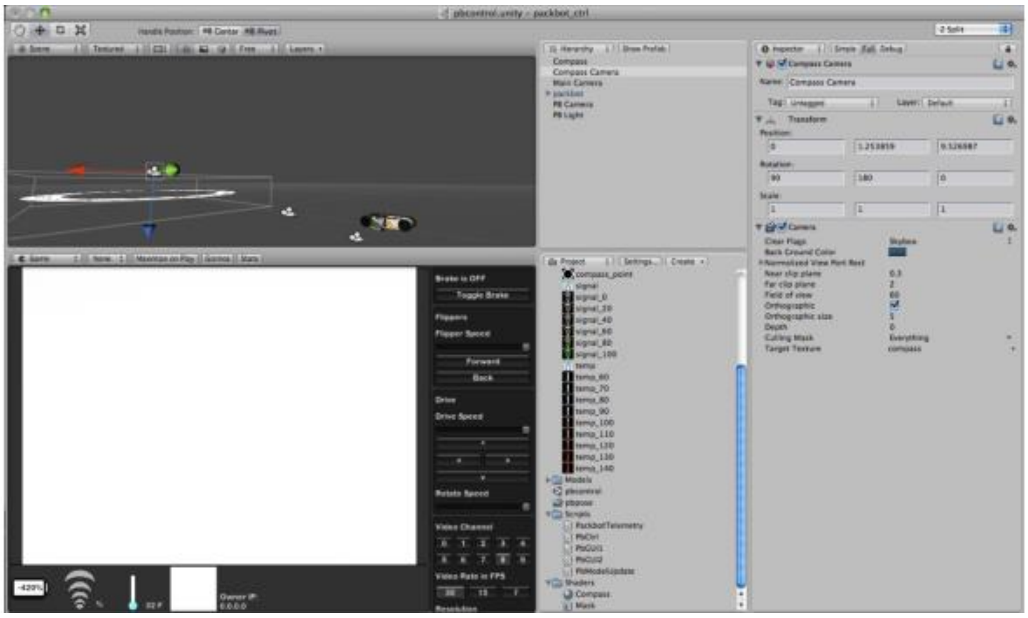


Figure 4.2: Unity game engine interface (Craighead et al., 2007)

Last but not least, the physics engine which is included, helps developers assign easily and realistically the laws of physics into the virtual environment. By just assigning the physics properties of an object (mass, drag, friction etc.) depending on the game developed, its desirable behaviour is simulated while all the complex calculations are automatically computed by the physics engine.

4.2.2 Qualcomm Vuforia SDK

Vuforia SDK is the tool that was used in order to implement the AR technology into the game. It is an SDK developed by Qualcomm which allows the execution of AR technology by handheld devices and smartphones equipped with a camera. Qualcomm is a company specialised on mobile devices' CPU therefore all of its products are highly optimised for these devices. Vuforia establishes marker-based AR technology by implementing on computer vision technology in order to track and recognise potential image-markers. This computer vision technology uses image tracking techniques that allows any image rich in natural features to be a marker expanding the AR technology into many areas. (Figure 4.3)

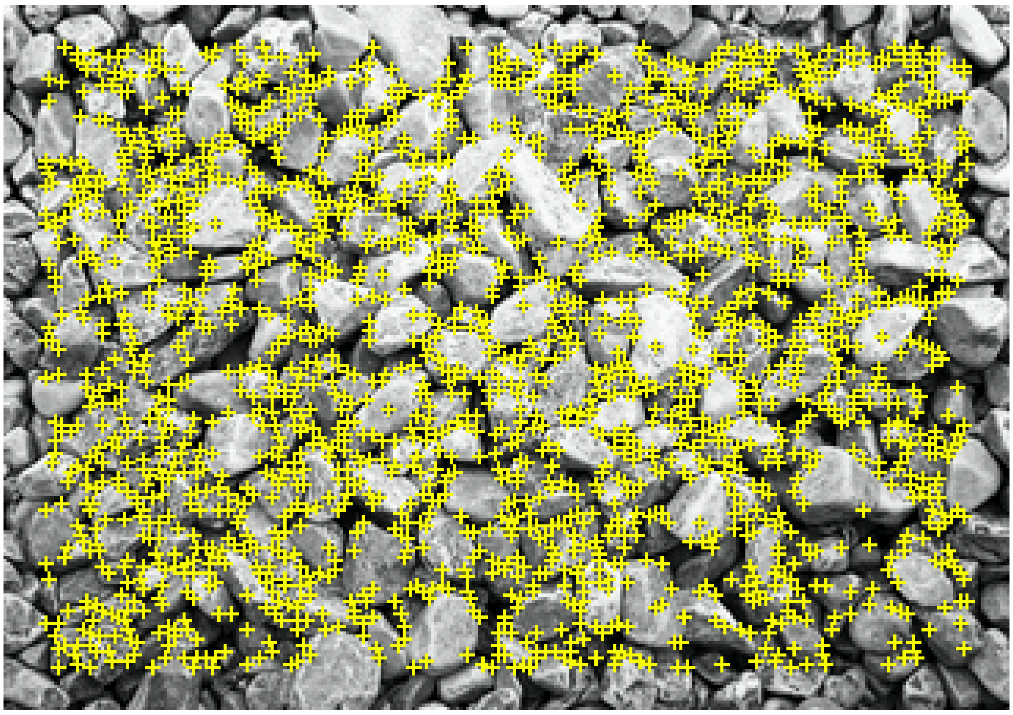


Figure 4.3: Tracking features on Vuforia marker-based image (Kim, & Kim, 2014)

Furthermore, one of the main reasons that Vuforia SDK was chosen, is its integration with Unity3D. A simple plugin is added into the engine importing everything that a developer will need to create an AR application. Previous experience with the specific tool (ARSoccer) showed that using the available components of the plugin allows fast development because most of the calculations are computed. Moreover, Vuforia contains many interesting features for developing a high-end AR application. Firstly, it supports multiple marker-based targets. These targets include images, cylinder, text and more allowing a big variety of possible implementation techniques. Next, virtual buttons feature opens new ways of interacting with an application. Virtual buttons are user-defined areas on the marker that when the users hovers their finger on that specific area an action is executed. Last, the tracking algorithm that is used allows fast recognition of a predefined (Drummond). The steps that algorithm follows are:

The image that the developer chooses to be a target is uploaded into the database target manager where its features are analysed by a corner detector and all the detected points are then stored into this database. These points are used for the tracking detection.

The video texture of the user’s camera uses the corner detector to analyse in real-time any possible points that match the ones predefined on the first step.

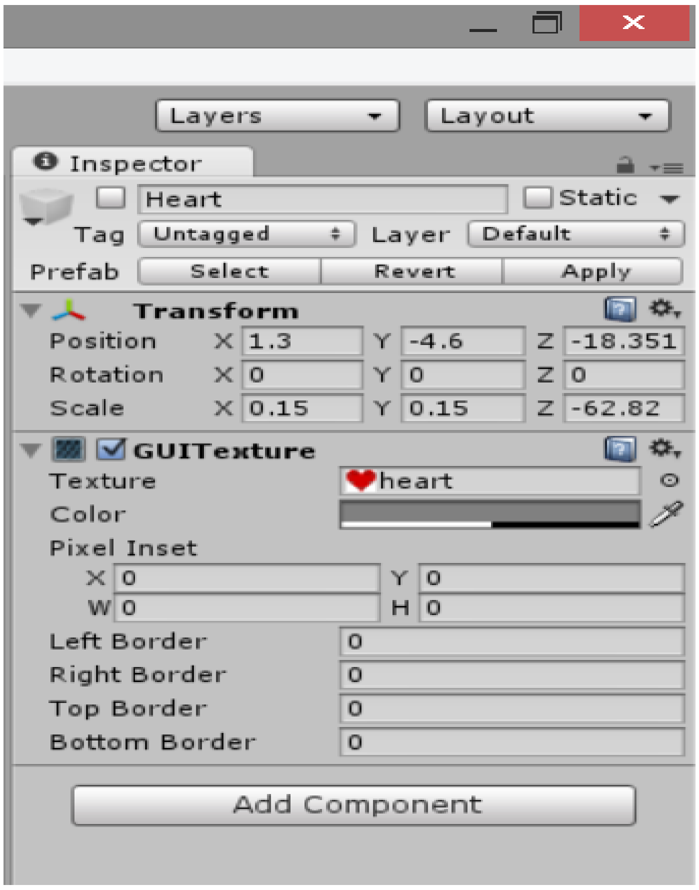
When a desirable number of points is detected and matched, a projective transformation matrix maps the points from the video texture with the ones of the predefined image while using error minimization methods.

Finally, from the matrix obtained before, the camera’s orientation and viewpoint is estimated and all the virtual objects, that are designed, are overlayed on the video texture at the correct positions.

4.3 2D Graphics – HUD

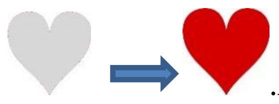
Using Unity3D, the aim is to create a Graphical User Interface that will be handy and at the same time attractive in order to be usable for all the available devices’ screen sizes. The GUI must be as simple as possible and familiar to the users so that they could easily adapt it and use it, without any difficulties and without extra instructions. Because the game uses AR technology and for that reason the device’s camera is used, the GUI must be designed in such a way that it will not block users’ main viewing and interacting screen.

Unity3D offers a very good GUI library that allows developers to create quickly and easily a big variety of GUI elements. This library automates most of the GUI elements’ functionality. Although, such a library seems the best choice for creating the HUD its performance on mobile devices is terrible. From previous experience, where GUI buttons were used for the main gameplay of a mobile game, the speed performance was below average making the game in some cases unplayable. The reason of this bad performance is because the OnGUI function is called multiple times per frame without even using it sometimes. To overcome this issue, GUITextures were used instead. GUITextures are simple images overlayed on top of the screen. Unity3D allows developers easily position, scale and change the image of the texture through the inspector (Figure 4.4).



```
Heart                Static
Tag     Untagged     Layer   Default
Prefab  Select  Revert  Apply

Transform
Position   X 1.3     Y -4.6    Z -18.351
Rotation   X 0       Y 0       Z 0
Scale      X 0.15    Y 0.15    Z -62.82

GUITexture
Texture        heart
Color
Pixel Inset
  X 0          Y 0
  W 0          H 0
Left Border    0
Right Border   0
Top Border     0
Bottom Border  0
```

Figure 4.4: Transform and GUITexture components on Unity Inspector

The positioning and scaling can be achieved either by the Pixel Inset option or through Transform properties. Between these two choices the best choice to be used is by changing the Transform properties and leaving blank the Pixel Inset properties. The reason is that when positioning and scaling is set by the Transform component, the Texture is positioned and scaled depending on the screen resolution of the device whilst Pixel Inset keeps the properties fixed. Moreover, any changes made, can be automatically seen on the editor which is very useful because developers gain valuable time. By now, the Textures are simple images overlayed on the screen without any extra functionality. In order to make these textures clickable, a script must be created. To achieve this, GUITexture class contains a function called HitTest which takes a position vector and returns true if that position is inside this Texture's margins. This allows checking when a touch input occurs, if it is inside those margins followed by the appropriate functionality that each Texture must perform. Furthermore, a way of showing the user that a GUITexture was pressed must be implemented. For that reason, a pair of the same images, but in different colour, for each button was placed so that when a texture is pressed, it disappears and appears the clicked state of it  .Although, this method requires developers to manually check when a texture is pressed rather than the automate way of GUI buttons, it dramatically optimises the performance.

After finding the most appropriate solution for designing the HUD, the sketches of all the possible screens were created. These screens are divided into four categories; create/ connect game, bidding round, main gameplay, counting point. Moreover, similar mobile applications for the specific game or for the French version of it (Belote) were reviewed (Figure 4.5).

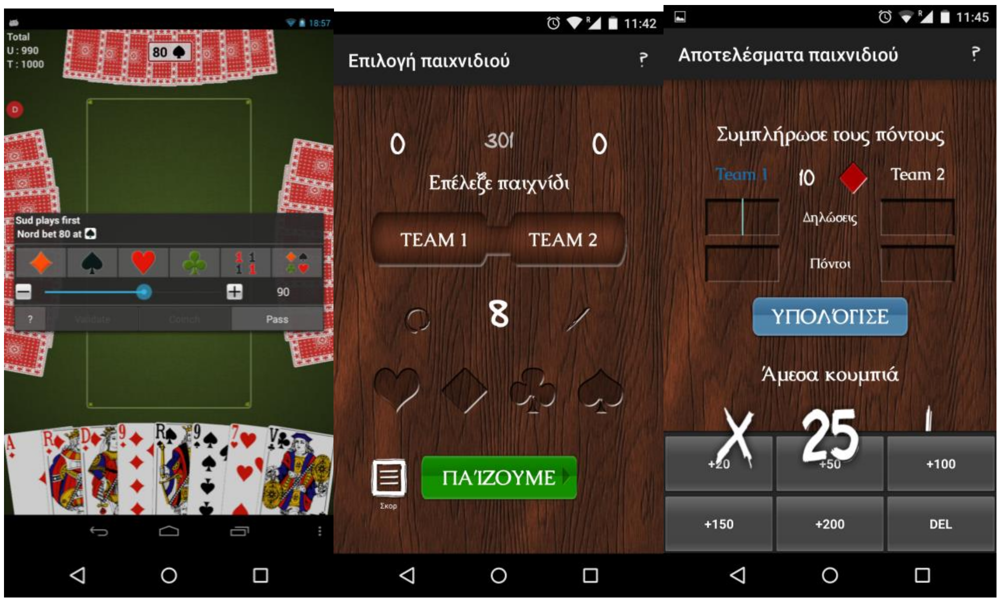


Figure 4.5: Belote Andr free (left) Repilo (middle, right)

The first screen is the one that appears when the app is launched prompting the players to start a new game. The first design included four buttons (GUITextures) as shown in Figure 4.6 where the players would press the appropriate button; first player should press “Player 1” button and so on.

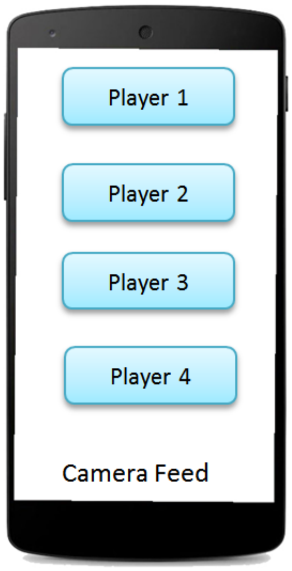


Figure 4.6: Initial design for Create/ Join game screen

After evaluating and reviewing this design, it was discarded because it was confusing and led to wrong presses because users did not know for sure which player button they should press. For that reason, a simpler and understandable design was created (Figure 4.7) with just two buttons. The first button indicates the creation of a new game and must be pressed by the player which was pre-decided to be the dealer. Then all the other three players must simply press the "connect" button to join the game created. When the first player creates the game a new button appears that shuffles and deals the cards to the players. This button is only clickable when all the players have joined the game.

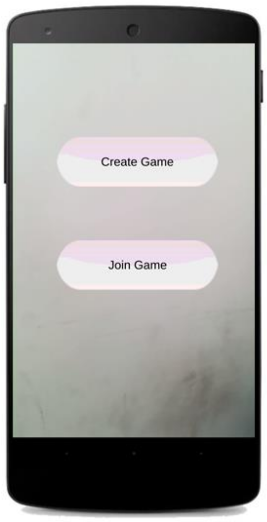


Figure 4.7: Final design for Create/ Join game screen

Next, when all the cards are dealt to the players the bidding round starts. Due to the fact that the bidding round is done in the same way as the traditional game where the players talk their bid, no extra buttons need to be implemented. The only GUI that must be designed is for the dealer who is going to declare the final bid of the round. Figure 4.8 shows the screen of the dealer with the appropriate textures.

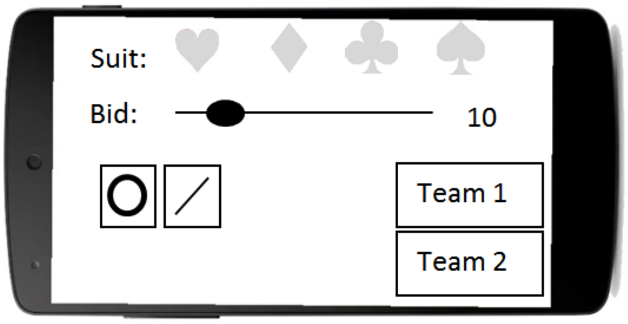


Figure 4.8: Initial design for bidding round screen

Firstly, the four card suits are situated on the top of the screen and below a slide bar to declare the final bid. Next, on the right side of the screen two buttons indicate the two teams, so the team that bid last must be pressed, and on the left side the optional buttons for "closed" and "double-closed". Finally, on the bottom right corner the play button is located, which becomes clickable when all the necessary options are chosen. After evaluating this design, users (dealers) have claimed two main issues. The first issue is that they could not see clearly their cards because all the textures were blocking them. For that reason, a "hide" button was implemented which when pressed hides all the textures of the screen allowing users to see their cards without any distractions. The second issue was with the slide bar. The slide bar was implemented to restrict the dealer with only the right variety of bid numbers (8-80) but users found it very difficult to choose the number they wanted because it was very sensitive. To overcome this issue a button implemented that opens the numeric keyboard of the device and checks whether the user's input is between the correct range. Figure 4.9 shows the final design of the dealer's screen.

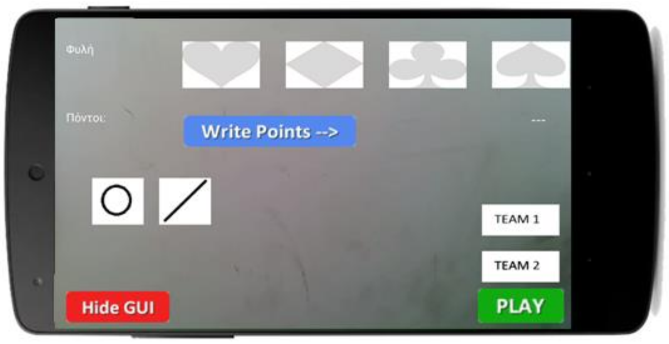


Figure 4.9: Final design for bidding round screen

The third screen is in the main gameplay where the actual game is executed. Here, the GUI must be as limited as possible so it will not block player's screen. Anyway, the only button that needs to be implemented is the one that will allow the players to show any declarations they might have, so a simple button is placed on the top right corner. The button's functionality will be discussed on User Interaction section.

The final screen appears only on dealer's screen when all the cards have been played in order to declare any extra points for each team from any declaration shown before. The initial design included two slide bars for each team (Figure 4.10).

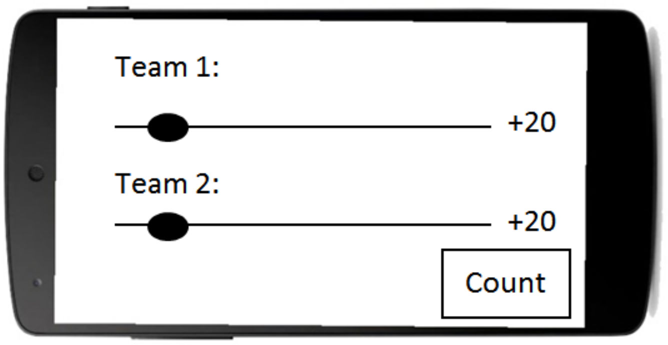


Figure 4.10: Initial design for counting round screen

Stating the issue on the bidding screen the design was decided to be implemented in a "calculator-based" style. Moreover, this game state is only for counting the points, so the issue of textures blocking the user's view does not affect the gameplay. As shown in Figure 4.11 the screen is divided into two parts indicating the two teams. For each team there are six buttons, the five specify the extra points and the sixth one is for clearing the points if a wrong touch press occurs. Finally, in the middle there is the "count" button which will sum the total points with the declarations.

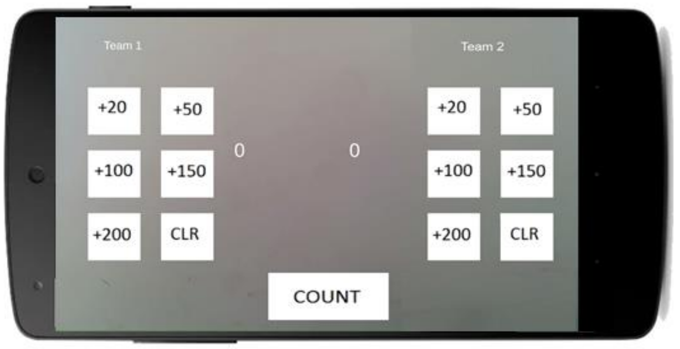


Figure 4.11: Final design for counting round screen

4.4 3D Graphics – Virtual Objects

Besides the Graphical User Interface, the game must contain the appropriate 3D objects that consists the general game space. As stated in previous sections, a card game was chosen to be implemented with AR technology because it will not require many virtual 3D objects that will make the game unrealistic. The main objective is to create an AR game environment that will feel and look as the traditional environment of that game.

The traditional game of Pilotta is played with 32 cards (from 7 to A) where the four players sit around a table accompanied with a paper where the dealer writes down the score. These three real objects (cards, table and paper) must be designed virtually in a way to be as realistic as possible.

Starting with the deck of cards, 32 cubes were created inside Unity3D. These cubes were scaled so that they will look as thin cards and the appropriate texture is applied on them. The texture as shown in Figure 4.12 seems very natural and is a well common texture that is found in many real cards.

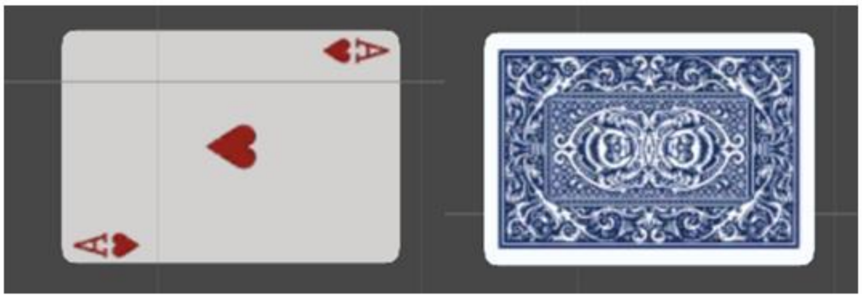


Figure 4.12: Ace of hearts gameobject with textures

Next, the paper needs to be created in a way that all four players will be able to view the score at any time. For that reason a big cube was developed and scaled appropriately to look like a real thin paper. A white texture was applied so that the virtual black text would be visible. Unity3D allows the creation of 3D text using the Text Mesh component so together with the white thin cube created, they consist a virtual “notebook”.

Finally, the table where all the cards will be thrown needs to be designed. At first, a big green plane was created with the appropriate physics where all the cards were on top. After reviewing this design, it was agreed that such a plane would look very unrealistic in such an environment. Therefore, it was decided to keep the big plane but remove its mesh renderer. This resulted to an invisible plane were all the physics are still available, tricking users that the real table, where the marker is situated on, is the one that stops the cards from falling.

4.5 Implementing Rules

4.5.1 Prototype Setup

The first step, for designing the final game, is to create a prototype where all the possible rules of the game are implemented. By firstly designing this prototype, it will ensure that the game will operate correctly before joining the AR technology and networking on it.

At first, the general game setup was created. This setup contains all the appropriate components for a virtual card game setup that can be played either by a pc or a mobile device. Starting with, a big green plane was created that indicates the floor while making all the other objects detectable. Next, a virtual camera is placed on top of that plane and rotated in order to view the plane. After that, the cards and the paper discussed in the previous section are placed on top of that plane and inside the Field of View of the camera. Furthermore, two empty gameobjects were created containing all the textures for the HUD. The first empty gameobject contains all the textures needed for the biding round and the other one all the textures needed for the final screen (counting points). Finally, another empty gameobject is created where all the general scripts needed for the game will be placed. Figure 4.13 shows the final prototype setup.

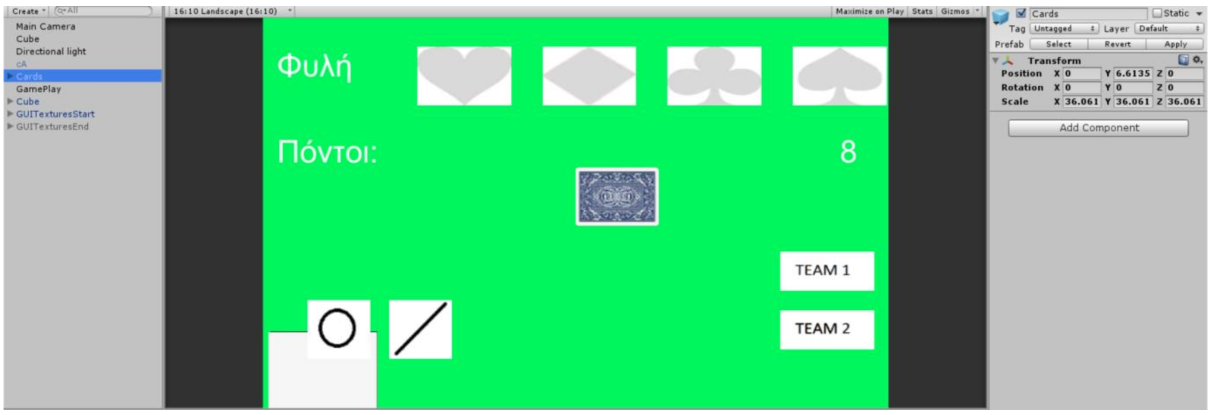


Figure 4.13: Final prototype setup in Unity3D

4.5.2 Card Values

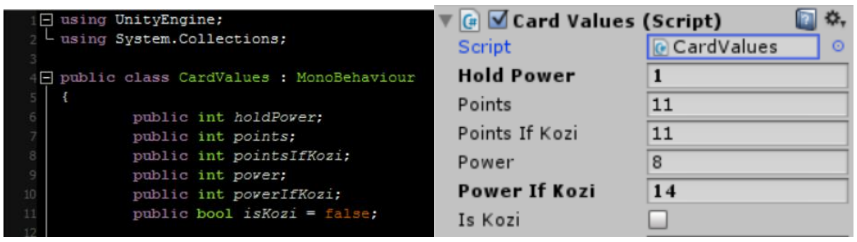
First of all, the 32 cards that were created must be named and marked properly. For that reason, a specific name was used for each card where the first character indicates the suit (c for clubs, h for hearts, s for spades and d for diamonds), followed by its value. For example, ace of spades is named as “sA”. Moreover, for the specific game, each card holds different power and different points when is indicated as trump card and different if it is not. Furthermore, a holding power is implemented that shows how the eight cards are traditionally hold by most players. Table 4.1 shows the power and points of each card in these two different situations.

Table 4.1: Power values and points of each card in all cases (*in parenthesis is shown the power over all the cards where outside of parenthesis the power between the 8 trump cards)

Card	A	K	Q	J	10	9	8	7
Holding Power	1	2	3	4	5	6	7	8
Power	8	6	5	4	7	3	2	1

Power if Trump*	6 (14)	4 (12)	3 (11)	8 (16)	5 (13)	7 (15)	2 (10)	1 (9)
Points	11	4	3	2	10	0	0	0
Points if Trump	11	4	3	20	10	14	0	0

To implement these rules, a script was attached to each of the 32 cards where six public variables are declared. These variables are declared public so that can be changed through the Inspector (Figure 4.14)



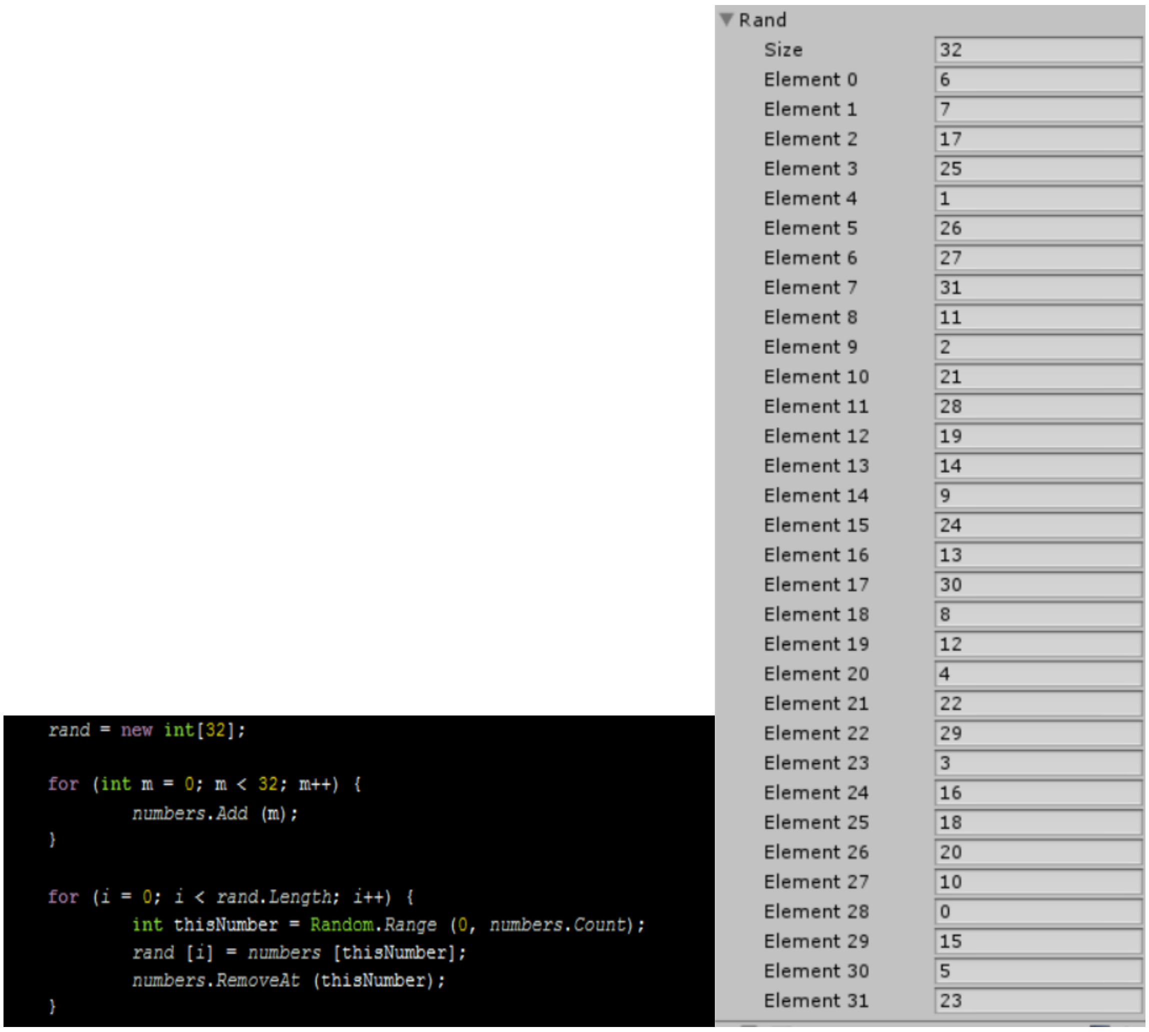
```
using UnityEngine;
using System.Collections;

public class CardValues : MonoBehaviour
{
        public int holdPower;
        public int points;
        public int pointsIfKozi;
        public int power;
        public int powerIfKozi;
        public bool isKozi = false;
```

Card Values (Script)	
Script	CardValues
Hold Power	**1**
Points	11
Points If Kozi	11
Power	8
Power If Kozi	**14**
Is Kozi	☐

Figure 4.14: Example code of CardValues script with its public variables (Kozi = Trump)

4.5.3 Shuffling and Dealing

The first thing that needs to be implemented is how the shuffling and the dealing of the cards to the four players will be generated. Initially a correct shuffling algorithm must be created. A list of integers is initialised and filled up with integers between 0 and 31. A new integer array is created with a size of 32. Then inside a for-loop a random number is generated between zero and the size the list created. This temporary number is then placed in the array and removed from the list so that it will not be occurred again from the random generator. When the loop is finished an array of 32 numbers randomly positioned is created (Figure 4.15).



```
rand = new int[32];

for (int m = 0; m < 32; m++) {
        numbers.Add (m);
}

for (i = 0; i < rand.Length; i++) {
        int thisNumber = Random.Range (0, numbers.Count);
        rand [i] = numbers [thisNumber];
        numbers.RemoveAt (thisNumber);
}
```

▼ Rand

Size	32
Element 0	6
Element 1	7
Element 2	17
Element 3	25
Element 4	1
Element 5	26
Element 6	27
Element 7	31
Element 8	11
Element 9	2
Element 10	21
Element 11	28
Element 12	19
Element 13	14
Element 14	9
Element 15	24
Element 16	13
Element 17	30
Element 18	8
Element 19	12
Element 20	4
Element 21	22
Element 22	29
Element 23	3
Element 24	16
Element 25	18
Element 26	20
Element 27	10
Element 28	0
Element 29	15
Element 30	5
Element 31	23

Figure 4.15: Example code of Shuffle function (left), example of shuffled array (right)

The next step after implementing the shuffling algorithm is the dealing algorithm that will deal the correct cards to the four players. Five gameobject lists are initialised, where the four represent the eight holding cards of each player that are going to be dealt and the fifth contains the 32 cards (deck). In the traditional game, the dealing of the cards is made in a standard three-round way. In the first round the dealer deals three cards to each player, then two cards and finally three cards. In order to keep this way of dealing into the virtual game the algorithm follows this three-round rule. Table 4.2 shows which cards, from the deck, each player must take.

Table 4.2: Cards of each player on dealing

	Round 1	Round 2	Round 3
Player 1	1, 2, 3	13, 14	21, 22, 23
Player 2	4, 5, 6	15, 16	24, 25, 26
Player 3	7, 8, 9	17, 18	27, 28, 29
Player 4	10, 11, 12	19, 20	30, 31, 32

This is implemented with four methods, one for each player, where all the cards are dealt to each player and tagged appropriately (all eight cards of player one are tagged as "player1" etc.). Moreover, the eight cards of each player are sorted depending on their holding power and their suit so they are arranged neatly and not confusing the players. When the dealing is done, all 32 cards must be placed in the correct positions according to the player's position. This positioning is used only for the specific prototype as for the final game the cards' positions will be different. Figure 4.16 shows how the positioning is done. All the cards are rotated so that they are visible to the camera.

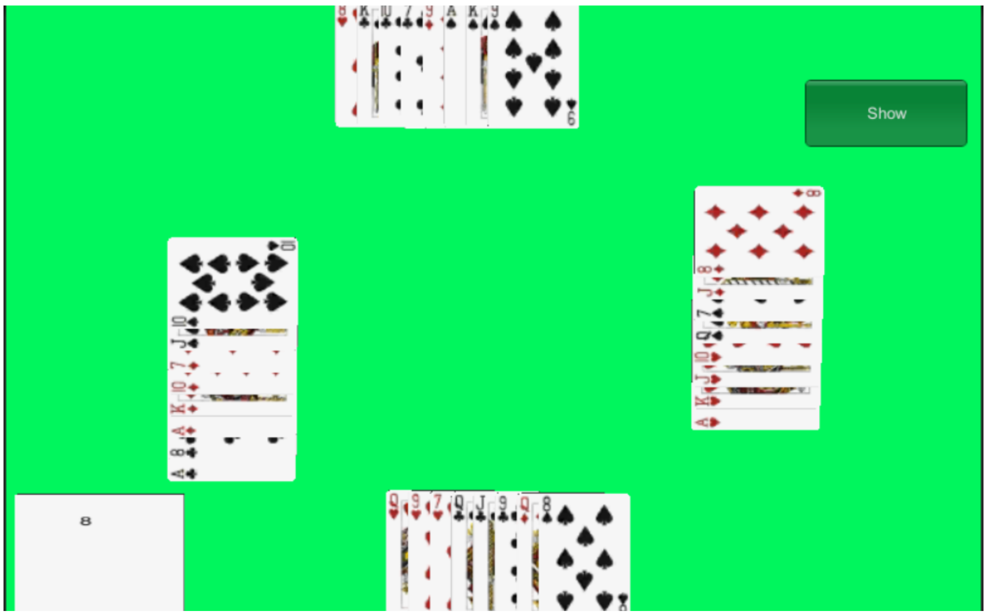
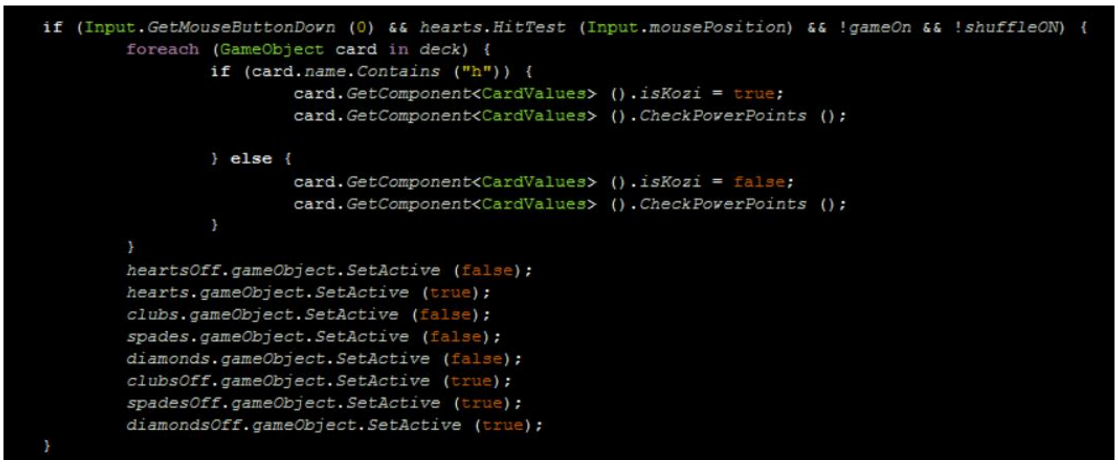


Figure 4.16: Cards positioning on first prototype

4.5.4 Bidding

In this section it is analysed the functionality of the GUI that was discussed in previous sections. In the bidding round the dealer must declare the final bid, the suit of the final bit, the team last bid and if the game was declared as “closed” or “double-closed”.

At first the dealer chooses the final suit between the four available. As soon as, a touch occurs, a check detects if the position of this touch input is on a suit texture. Then, if the dealer chose a suit texture, a method finds all the cards of that suit; for example if hearts are selected, it searches all the 32 cards that contain the character “h” in their name. When all the correct cards are found, another method accesses their CardValues script to change their power and points to the appropriate value (Figure 4.17), marks them as “Trumps” and changes the button textures in order to seem like the specific texture is selected.



```
if (Input.GetMouseButtonDown (0) && hearts.HitTest (Input.mousePosition) && !gameOn && !shuffleON) {
        foreach (GameObject card in deck) {
                if (card.name.Contains ("h")) {
                        card.GetComponent<CardValues> ().isKozi = true;
                        card.GetComponent<CardValues> ().CheckPowerPoints ();

                } else {
                        card.GetComponent<CardValues> ().isKozi = false;
                        card.GetComponent<CardValues> ().CheckPowerPoints ();
                }
        }
        heartsOff.gameObject.SetActive (false);
        hearts.gameObject.SetActive (true);
        clubs.gameObject.SetActive (false);
        spades.gameObject.SetActive (false);
        diamonds.gameObject.SetActive (false);
        clubsOff.gameObject.SetActive (true);
        spadesOff.gameObject.SetActive (true);
        diamondsOff.gameObject.SetActive (true);
}
```

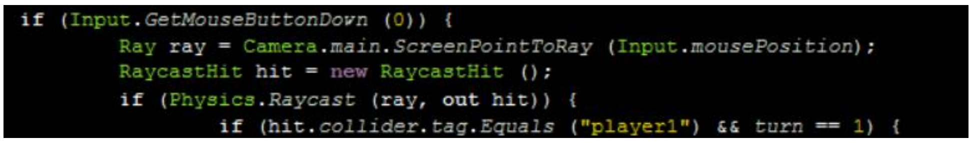
Figure 4.17: Example code of selecting the “hearts” texture

Next, the dealer declares the final bid through the numeric keyboard of its device. This final bid is then saved into a variable which is going to be used to determine the winner after the specific round. Also, the final bid must be between 8 and 80 so if the dealer accidentally declares a different number, the play button remains deactivated until the dealer corrects it. The same procedure is followed for all the other buttons (last team bid, “closed” and “double-closed”) because these values will be needed on the “counting score” round.

4.5.5 Main gameplay

The main gameplay starts when the dealer declares the game and its rules are broken down into three categories. The first category deals with what cards the players can throw to handle any wrong throwing, the second is how the user would show any of their declarations, and the third deals with who the winner of each round should be.

Starting with the first category the script must determine which cards a player is allowed to throw. The interaction with the cards implemented in this prototype is different from the final prototype because here the players can choose a card by simply touching it. The final user interaction will be explained in the following chapters. For this prototype, a ray is casted from the position of the camera to the position of the player's touch input. As stated before, the game is played in a turn-by-turn way where the next player must play only when the players before him/ her have thrown their cards. So, this ray checks if it hits any of the cards gameobject while checking whose turn is. For example, when it is the first player's turn (turn = 1) the ray checks if the hit object is tagged as "player1" in order to start the further checking (Figure 4.18).



```
if (Input.GetMouseButtonDown (0)) {
        Ray ray = Camera.main.ScreenPointToRay (Input.mousePosition);
        RaycastHit hit = new RaycastHit ();
        if (Physics.Raycast (ray, out hit)) {
                if (hit.collider.tag.Equals ("player1") && turn == 1) {
```

Figure 4.18: Example code of checking if the ray casted hits a card with "player1" tag

When the ray hits a card of the appropriate player, a function is called (ThrowCard(hit, pos)) which gets as variables the hit gameobject and a particular position where the card should be placed. As explained in the Rules section, the player who plays first can throw any card from his/her eight cards. This card's suit is placed into a temporary string (firstPlayed) variable in order to be used as a check for the next players. Finally, the card is placed into an array of four slots (CardsOnTable), it takes the position passed in the function and the turn is increased so the next player can play (Figure 4.19).

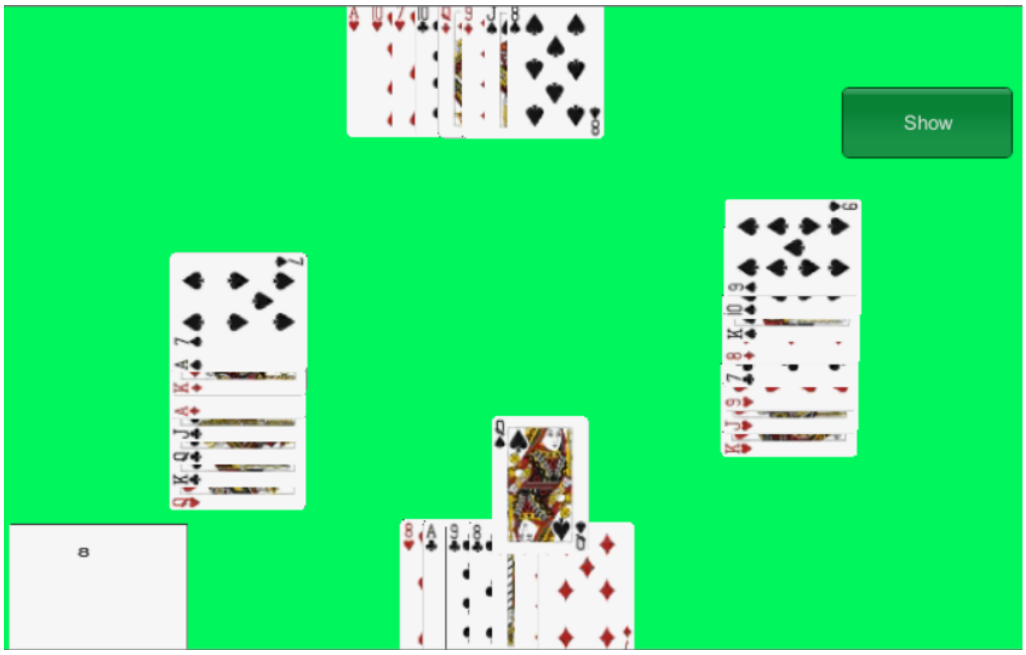
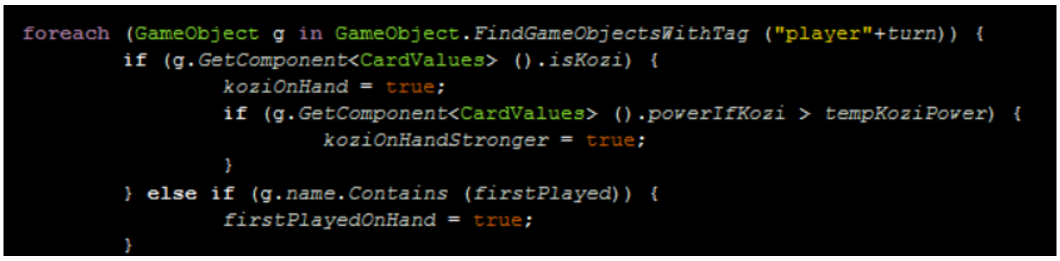


Figure 4.19: Position of a thrown card

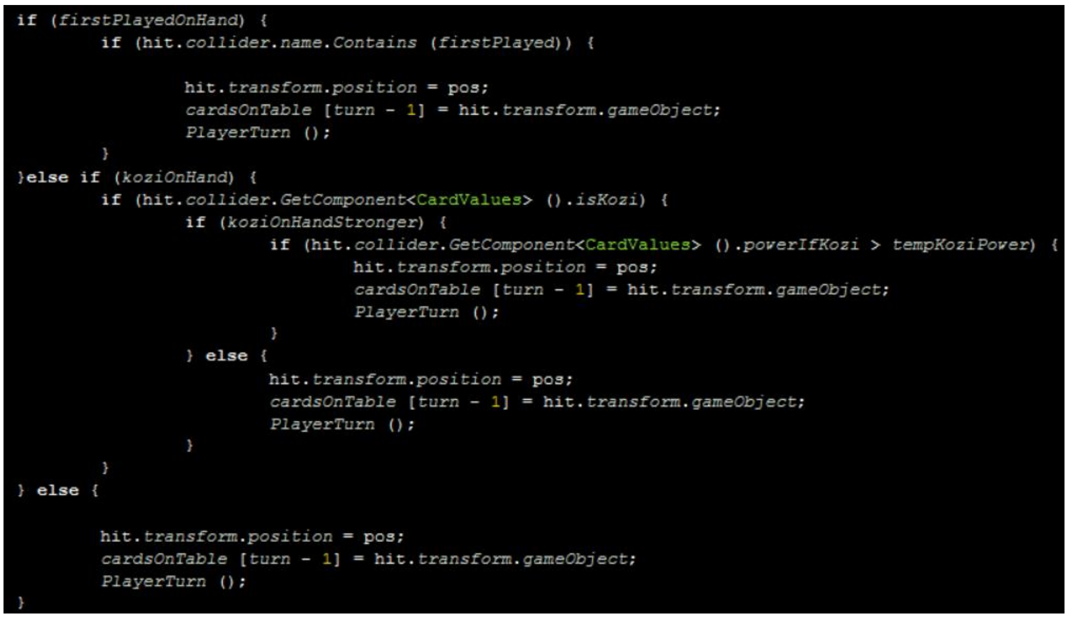
Above, is explained the first player's turn where no major rules are valid because he/ she can throw any card, either trump card or not. Now, for implementing correctly the next players' turn further checks are needed inside the ThrowCard function. Firstly, inside the Update() function, as soon as a gameobject is passed into the CardsOnTable array a check determines if any of the cards inside that array is a trump card and holds the highest trump power into an integer variable (tempKoziPower). Then, when the next player clicks on a card, inside the ThrowCard function, a foreach statement checks all the cards of that player and determines whether three possible scenarios occur. The first scenario is whether the player holds any cards with the same suit as the card that was first played (firstPlayedOnHand). The second checks if the player holds any trump card (koziOnHand), where the third scenario checks if any of these holding trump cards is higher from the tempKoziPower (koziOnHandStronger). When any of these three scenarios occur a respective Boolean variable becomes true (Figure 4.20).



```
foreach (GameObject g in GameObject.FindGameObjectsWithTag ("player"+turn)) {
        if (g.GetComponent<CardValues> ().isKozi) {
                koziOnHand = true;
                if (g.GetComponent<CardValues> ().powerIfKozi > tempKoziPower) {
                        koziOnHandStronger = true;
                }
        } else if (g.name.Contains (firstPlayed)) {
                firstPlayedOnHand = true;
        }
```

Figure 4.20: Example code for checking player's cards values

Subsequently, as mentioned in Rules section, a sorted if-statement, with the three Boolean variables explained above, chooses if the card selected can be thrown or not. The sorting based on the rules is shown in Figure 4.21.



```
if (firstPlayedOnHand) {
        if (hit.collider.name.Contains (firstPlayed)) {

                hit.transform.position = pos;
                cardsOnTable [turn - 1] = hit.transform.gameObject;
                PlayerTurn ();
        }
}else if (koziOnHand) {
        if (hit.collider.GetComponent<CardValues> ().isKozi) {
                if (koziOnHandStronger) {
                        if (hit.collider.GetComponent<CardValues> ().powerIfKozi > tempKoziPower) {
                                hit.transform.position = pos;
                                cardsOnTable [turn - 1] = hit.transform.gameObject;
                                PlayerTurn ();
                        }
                } else {
                        hit.transform.position = pos;
                        cardsOnTable [turn - 1] = hit.transform.gameObject;
                        PlayerTurn ();
                }
        }
} else {

        hit.transform.position = pos;
        cardsOnTable [turn - 1] = hit.transform.gameObject;
        PlayerTurn ();
}
```

Figure 4.21: Example code for allowing players throw the correct card

When the correct card is selected by the players it is placed in the CardsOnTable array, it takes the appropriate played position and the player's turn is increased.

Continuing on the second category, the players must be able to show any declarations in their cards to the other players. These declarations give extra points to the player's team. Table 4.3 shows these declarations with their points.

Table 4.3: Declaration points

Declaration	Extra Points
Four J's	200
Four 9's	150
Four A's	100
Four 10's	100
Four K's	100
Four Q's	100
Five sequentially cards of the same suit	100
Four sequentially cards of the same suit	50
Three sequentially cards of the same suit	20

In the traditional game, the players, in the first round of the gameplay, say their declarations and in the second round, if they do not forget or otherwise, show the declaration to the other players. In the mobile game prototype a way of showing these cards needs to be developed. For that reason, a button is implemented which is situated in the top right corner of each player's screen. When they click that button the user interaction changes its state from "throw-card" state into "show-card" state. In that way, instead of calling the ThrowCard function a ShowDeclaration function is called with the same passing variables. Hence, each card hit by the ray is placed into a DeclaredCards array and its position is changed to be visible by all the other players (Figure 4.22). If a player accidentally clicks a wrong card, he/ she can click it again to bring it back to its original position.

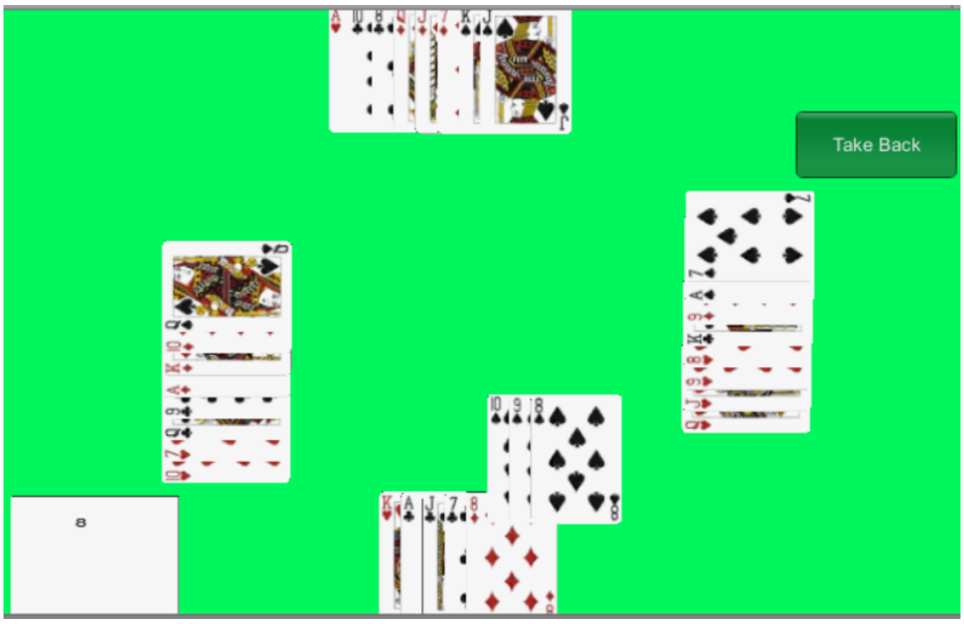
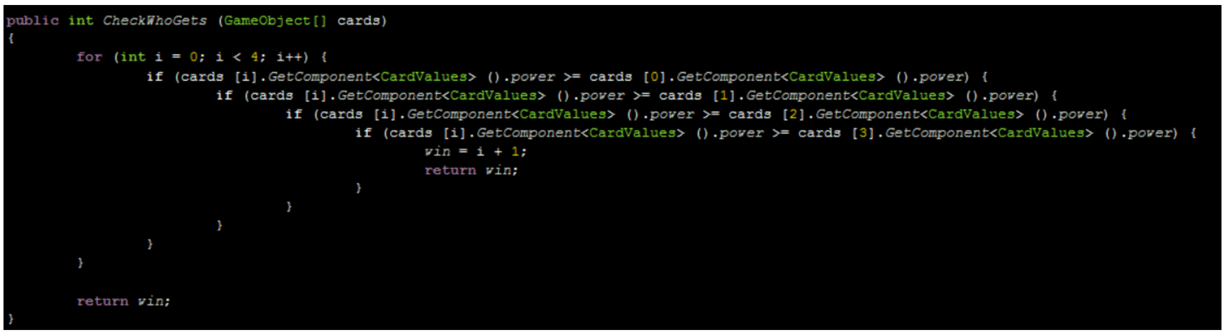


Figure 4.22: Showing a declaration on first prototype

When the player has shown the declaration, it clicks the "Take Back" button which clears the DeclaredCards array, takes all the showing cards back to their original position and returns back to the "throw-card" state.

Finally, the third scenario must determine the winner on each of the eight rounds of the main gameplay. It occurs when all four players have played their card and needs to check who has thrown the higher card on that round. Inside the Update() method an if-statement checks whether the CardsOnTable array has four elements which will indicate that all four players have played. Then, a function is called from the Rules script (Figure 4.23), passing the array as a variable, to determine the winner. Inside that function, a for-statement takes each of the four cards and compares its power with the other three. If a card's power is bigger than all the other three it returns the winning card's position which indicates the player who has thrown it. For example if the highest card was placed in the third slot of the CardOnTable array then the winner of that round is Player 3.



```
public int CheckWhoGets (GameObject[] cards)
{
        for (int i = 0; i < 4; i++) {
                if (cards [i].GetComponent<CardValues> ().power >= cards [0].GetComponent<CardValues> ().power) {
                        if (cards [i].GetComponent<CardValues> ().power >= cards [1].GetComponent<CardValues> ().power) {
                                if (cards [i].GetComponent<CardValues> ().power >= cards [2].GetComponent<CardValues> ().power) {
                                        if (cards [i].GetComponent<CardValues> ().power >= cards [3].GetComponent<CardValues> ().power) {
                                                win = i + 1;
                                                return win;
                                        }
                                }
                        }
                }
        }

        return win;
}
```

Figure 4.23: CheckWhoGets function

When the winner is returned, a coroutine is started which moves the four cards to the winner’s side to indicate the winner in the same way as the traditional game. After the cards are placed in the winner’s side all the variables used for that round are reset to their default value, the CardOnTable array is cleared and the turn is changed to the winner player who must play first on the next round. Lastly, a final check determines if it is the eighth round of the main gameplay in order to add ten extra points to the team who won that last round.

4.5.6 Counting the points

In this section is explained how the counting of the points is calculated and also how any declarations shown in the main gameplay are added to that total.

Starting with, the counting of the points is calculated at the end of each round inside the main gameplay. Besides of checking who the winner of each round is, as explained above, a function is implemented to count the points of the four cards played. The function takes the CardOnTable array and sums the points of the four cards and, depending on who the winner is, adds that sum to the respective team. For example, if the winner is the player one or the player three the sum is added to team one, else on team two.

When all eight rounds are played, in the dealer’s device appears the “Add-Declarations” screen as explained in HUD section. When the dealer clicks on the buttons with the extra points, these extra points are added into an integer variable that holds the declaration points of each team. After declaring the declarations the dealer clicks the “Count” button to add these declarations to the total of each team and determine the winner. The winner is decided when the team that bid last on the bidding round gets the appropriate points. For example, team one had bid last and their bid was 10 of spades. Then, team one must win 100 or more points in order to win the game, or else team two gets the appropriate points. Finally, when the total of each team is calculated, it is written on the paper gameobject as shown in Figure 4.24.

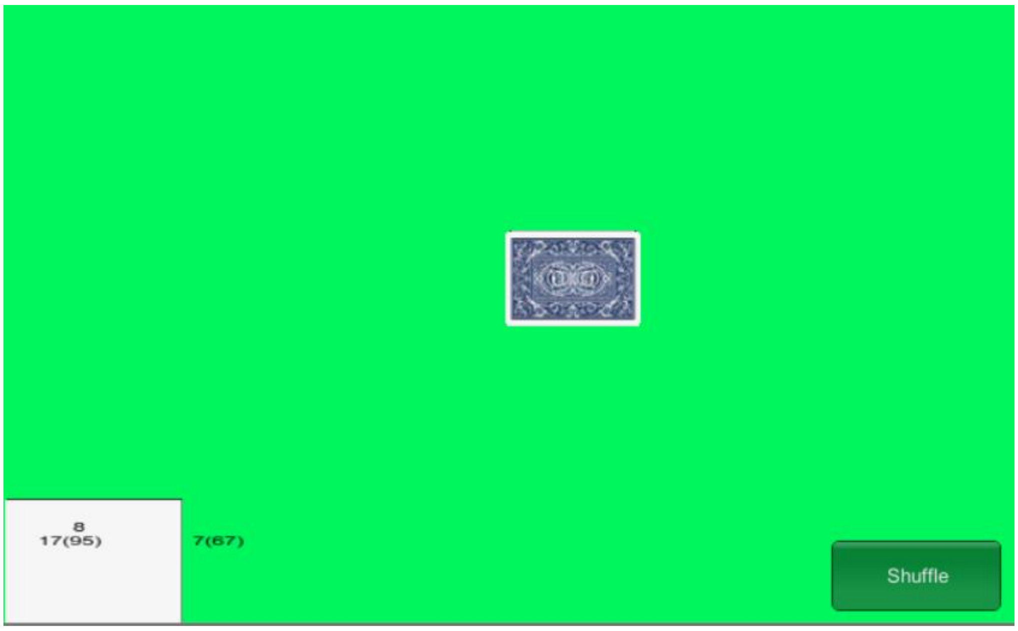


Figure 4.24: Displaying the score on first prototype

4.6 Prototype Evaluation

After the prototype was implemented an evaluation was run to determine any possible errors. For this evaluation, a big number of possible gameplay scenarios were created to test every aspect of that prototype.

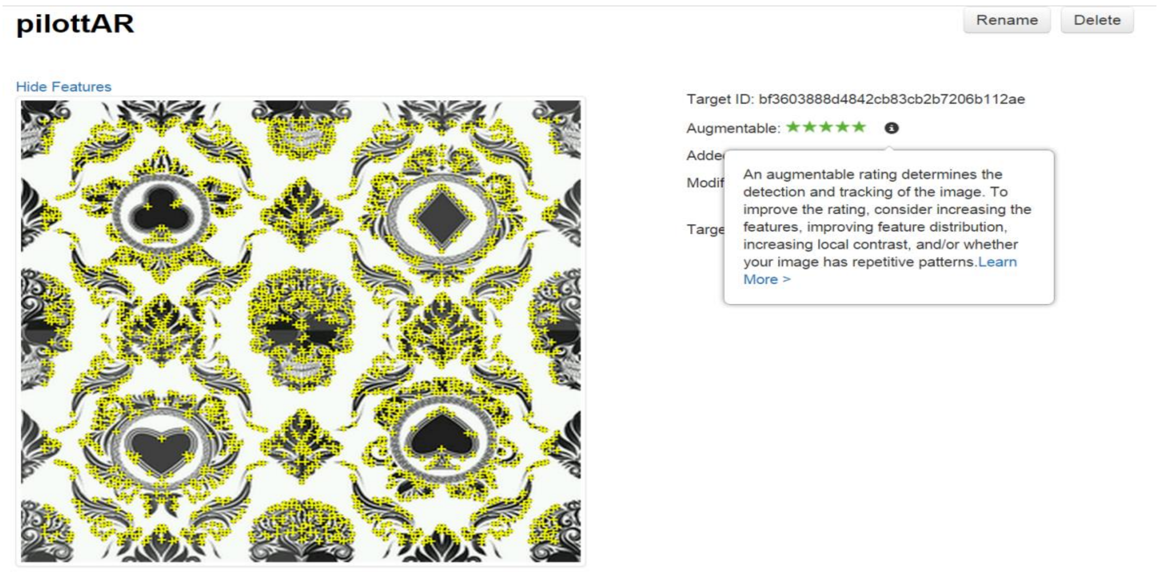
At the end of the evaluation various issues were determined and corrected. The first issue occurred in the scenario of “throwing any other card if the player does not have a card with the same suit as the first played and also he/ she does not have any trump cards”. The specific card does not have any power over the other three cards even if it is a high power card. For example, the first three players threw 8, 9 and 10 of spades respectively, and the fourth player because he/ she does not have any spades or trumps throws the ace of hearts. The system determines that the ace of hearts is the higher but actually the rules say that the winner is 10 of spades because it is the suit that was played first. For that reason, an extra code snippet was added to reset the power of that card to zero in order to not affect the check which determines the winner. The other issues were concerned with the calculations made on counting the total points, which were reviewed and corrected.

4.7 Implementing Augmented Reality

This section will describe how AR technology was implemented into the prototype described above. In the prototype, the game setup is in 2D so for merging the AR technology everything needs to be in 3D. All the scripts developed on the prototype are used, with minor changes, to provide the GUI and the rules of the game.

4.7.1 Unity project Setup

The first thing that needs to be done is to choose what marker is going to be used. For this project, an image target was chosen. In order to make this image an AR marker-based target, it needs to be uploaded into the target manager on vuforia's developer site. A developer can create a database and upload all of his/ her images that he/ she wants to have as markers. When an image is uploaded, it is then analysed and an augmentable rating based on its features is calculated. This rating expresses how rich in features is an image so, higher rating means better recognition by the device. Figure 4.25 shows the image that was chosen for PilottAR.



Target ID: bf3603888d4842cb83cb2b7206b112ae

Augmentable: ★★★★★

Adde

Modif

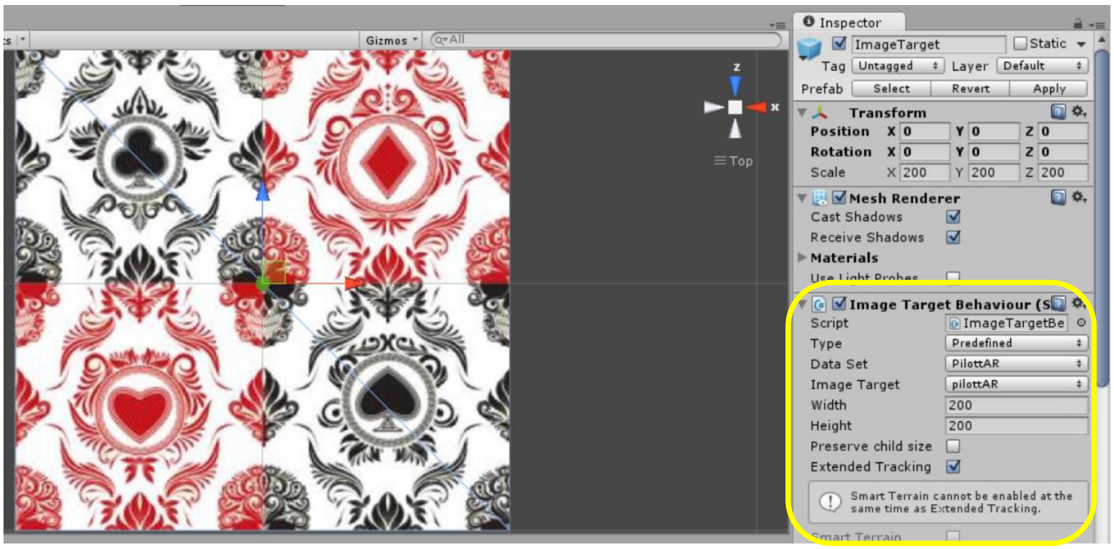
Targe

An augmentable rating determines the detection and tracking of the image. To improve the rating, consider increasing the features, improving feature distribution, increasing local contrast, and/or whether your image has repetitive patterns.Learn More >

Figure 4.25: The high augmentability image chosen as marker

This image is then downloaded as a unity package and added together with the vuforia plugin into the Unity project's prototype.

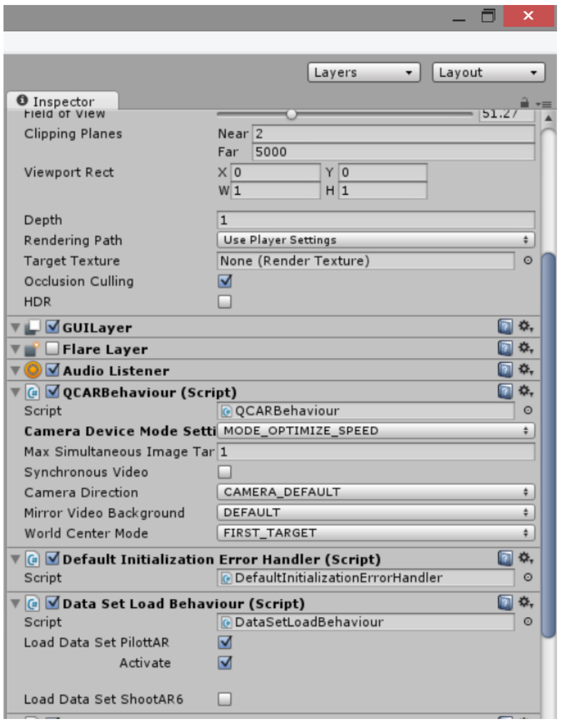
Next, the ImageTarget gameobject is added into the scene. This gameobject contains all the scripts that are needed to augment the elements on that image. The texture of that object is changed into the image that was previously chosen (Figure 4.26). Inside this gameobject, all the 3D elements described in section 4.4 are added as children.



✓ ImageTarget					☐ Static	
Tag	Untagged		Layer	Default		
Prefab	Select		Revert		Apply	
Transform						
Position	X	0	Y	0	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	200	Y	200	Z	200
✓ **Mesh Renderer**						
Cast Shadows	✓					
Receive Shadows	✓					
Materials						
Use Light Probes	☐					
✓ **Image Target Behaviour (S**						
Script	ImageTargetBe					
Type	Predefined					
Data Set	PilottAR					
Image Target	pilottAR					
Width	200					
Height	200					
Preserve child size	☐					
Extended Tracking	✓					
Smart Terrain cannot be enabled at the same time as Extended Tracking.						
Smart Terrain	☐					

Figure 4.26: ImageTargetBehaviour script component in Inspector

Furthermore, some changes need to be done in order to run on a mobile device using the device's camera. As mentioned before, Vuforia plugin contains all the appropriate elements to create easily an AR application. The Main Camera object needs to be replaced with the ARCamera object of Vuforia. This ARCamera gameobject provides by prior all the scripts needed for the correct behaviour of the camera. As shown in Figure 4.27, on the Data Set Load Behaviour script the data set of PilottAR needs to be selected. This will activate the recognition of the texture that is inside PilottAR data set, downloaded before. Vuforia allows multiple data sets to be activated which means that a developer can use multiple images to execute the same functionality, in our case the card game.



```
Field of View                          51.27
Clipping Planes                        Near 2
                                       Far  5000
Viewport Rect                          X 0        Y 0
                                       W 1        H 1
Depth                                  1
Rendering Path                         Use Player Settings
Target Texture                         None (Render Texture)
Occlusion Culling                      ✓
HDR                                    ☐
✓ GUILayer
☐ Flare Layer
✓ Audio Listener
✓ QCARBehaviour (Script)
Script                                 QCARBehaviour
Camera Device Mode Setti               MODE_OPTIMIZE_SPEED
Max Simultaneous Image Tar             1
Synchronous Video                      ☐
Camera Direction                       CAMERA_DEFAULT
Mirror Video Background                DEFAULT
World Center Mode                      FIRST_TARGET
✓ Default Initialization Error Handler (Script)
Script                                 DefaultInitializationErrorHandler
✓ Data Set Load Behaviour (Script)
Script                                 DataSetLoadBehaviour
Load Data Set PilottAR                 ✓
                    Activate           ✓
Load Data Set ShootAR6                 ☐
```

Figure 4.27: DataSetLoadBehaviour script component in Inspector

After the setup is created the elements are augmented as shown in Figure 4.28 and the hardware used for testing the AR implementation is a Nexus 5 smartphone.

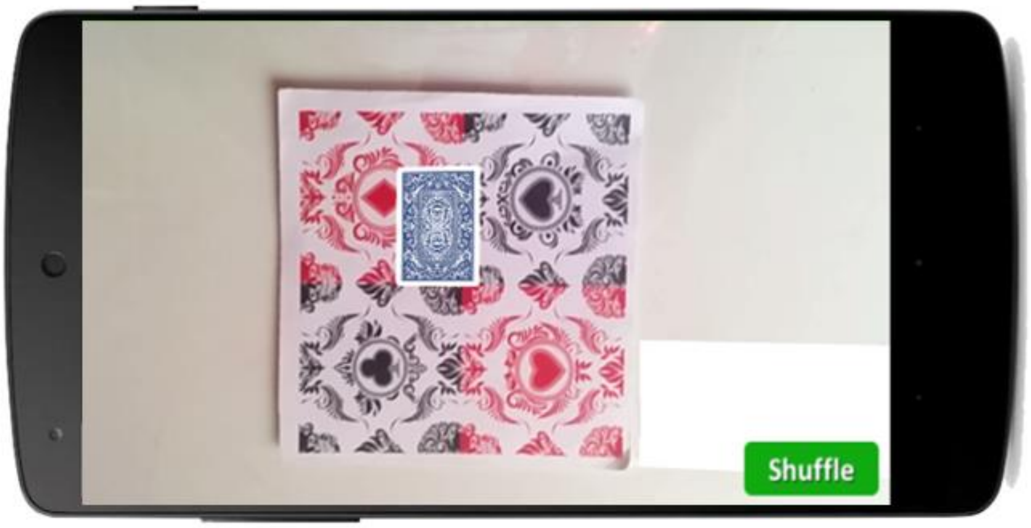


Figure 4.28: Augmenting the objects through the device

4.7.2 Game's Environment Setup

In the prototype the developed cards of the four players are placed in way to be visible from the static camera. For implementing the AR that will use the camera of the four players' device, another setup needs to be developed. This setup must be developed in a way that each player can see only his/ her cards and eliminate cheating as much as possible. The initial design as shown in Figure 4.29 was to place the eight cards of each player in front the camera's device so whenever the user moves the device, the cards to remain visible.

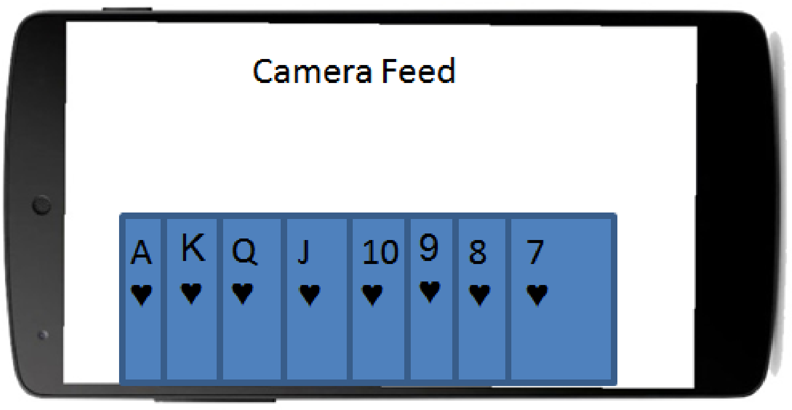


Figure 4.29: Initial design of players' cards position

After evaluating this design, some issues were found and the design was discarded. The main reason that the design was discarded was that the cards were blocking the view of the player. The viewing space was very limited and in some cases the user could not see and track the marker. At first, to overcome this issue, the cards were resized and made smaller but then the interaction with the cards was more difficult because the distance between the cards were too small and led to fault touching. Moreover, the orientation of the device must be in landscape something that limits the users' freedom to use their device in the way they prefer.

For that reason a new design was created. This new design shown in Figure 4.30 was inspired by board games like Scrabble where players have their own pad by their side. The evaluation of this design showed that, although, the cards were static, the players could place their device's camera anywhere they like while keeping the viewing space clear.

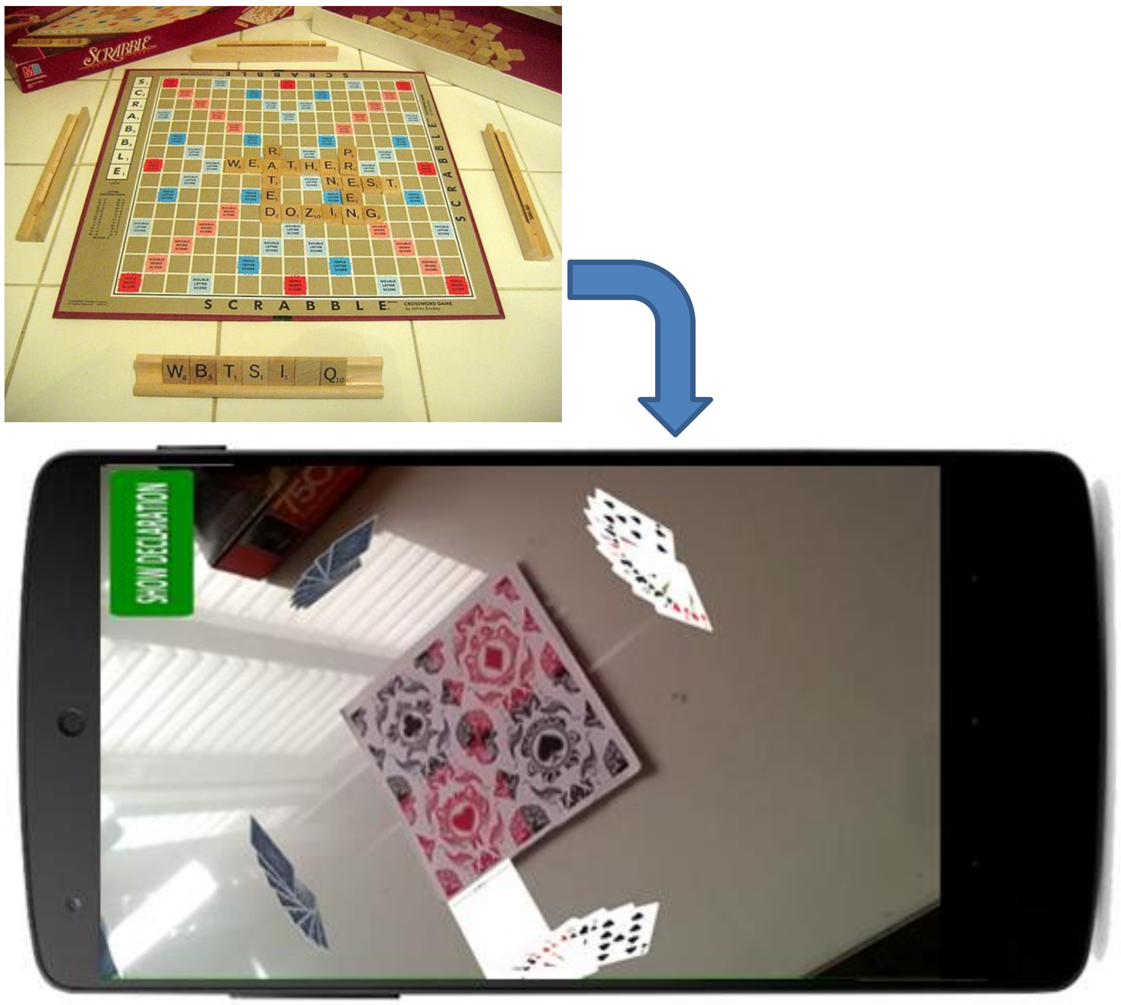


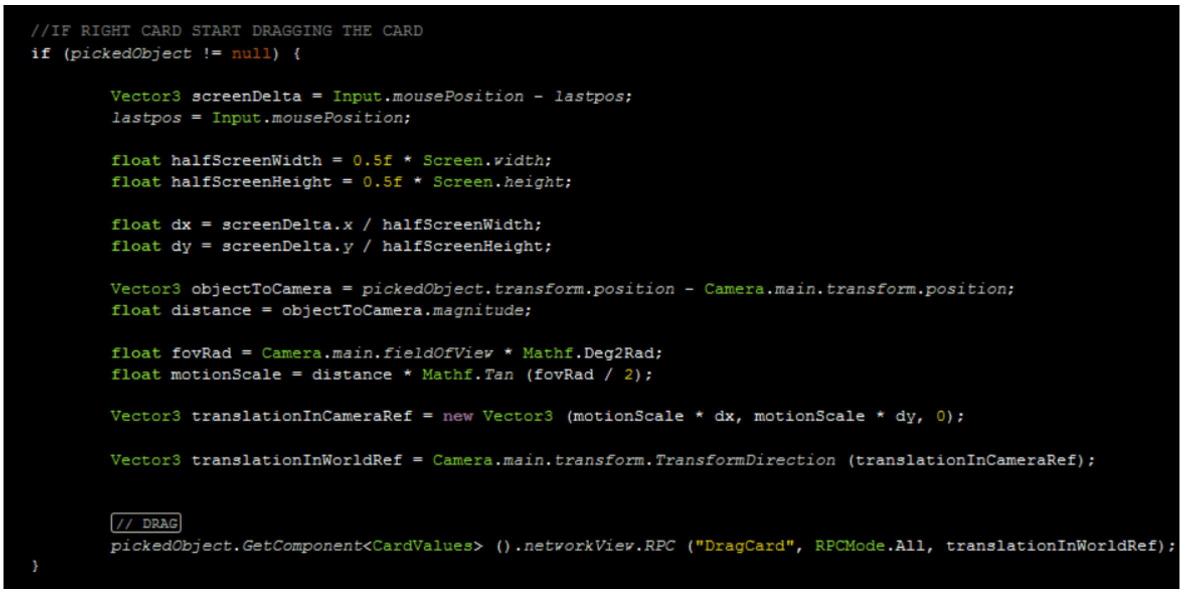
Figure 4.30: Final design of players' cards position inspired from Scrabble

4.7.3 User Interaction

In this section it will be described how users are interacting with the cards in the main gameplay. This interaction must be different from the prototype in order to handle any incorrect play.

In the prototype the players were selecting their cards by simply touching them. This could lead on choosing a card that they did not actually wanted to throw. To avoid this, another way of interaction should be implemented. Besides that, this new interaction must be mimicking the traditional way of throwing the cards to increase the immersion of the players and trick them, as much as possible, that they are actually holding and throwing real cards.

The interaction decided to be implemented as a “drag-n-throw” method. This way is very similar to the real play where the players select a card and throw it to the table. Firstly, the interaction script must be changed to this new method. When a user touch down occurs, a ray is casted from that input position and checks whether it hits to any appropriate card as discussed before. Then, when a correct card is selected, this card is saved into a temporary variable (pickedObject). With the proper calculations shown in Figure 4.31 the card selected can be dragged anywhere the user’s touch is. When the players release their finger from the screen, the card is thrown towards the table. This process implements the “drag-n-throw” method.



```
//IF RIGHT CARD START DRAGGING THE CARD
if (pickedObject != null) {

        Vector3 screenDelta = Input.mousePosition - lastpos;
        lastpos = Input.mousePosition;

        float halfScreenWidth = 0.5f * Screen.width;
        float halfScreenHeight = 0.5f * Screen.height;

        float dx = screenDelta.x / halfScreenWidth;
        float dy = screenDelta.y / halfScreenHeight;

        Vector3 objectToCamera = pickedObject.transform.position - Camera.main.transform.position;
        float distance = objectToCamera.magnitude;

        float fovRad = Camera.main.fieldOfView * Mathf.Deg2Rad;
        float motionScale = distance * Mathf.Tan (fovRad / 2);

        Vector3 translationInCameraRef = new Vector3 (motionScale * dx, motionScale * dy, 0);

        Vector3 translationInWorldRef = Camera.main.transform.TransformDirection (translationInCameraRef);


        // DRAG
        pickedObject.GetComponent<CardValues> ().networkView.RPC ("DragCard", RPCMode.All, translationInWorldRef);
}
```

Figure 4.31: Example code of dragging a selected card

Furthermore, an extra mechanism must be developed so the incorrect play to be avoided. A scenario was created to make the process understandable. In this scenario, the turn is on the fourth and last player of the round and on the table the following cards were thrown by the other three players respectively: 8, 9 and 10 of spades. The fourth player holds the 7 and Ace of spades. In order to win the hand, the player must throw the Ace. In his rush, the player selects the 7 of spades but before throwing it he realises his mistake and puts the card back and selects and throws the Ace. This error handling was implemented by checking the height distance between the first touch position and the position where this touch was released. When this distance is over the threshold the card is released, otherwise is placed back to its previous position. Thus, when a player selects an incorrect card, he can drag it back and select the correct one. The scenario described above is implemented and is shown in Figure 4.32.

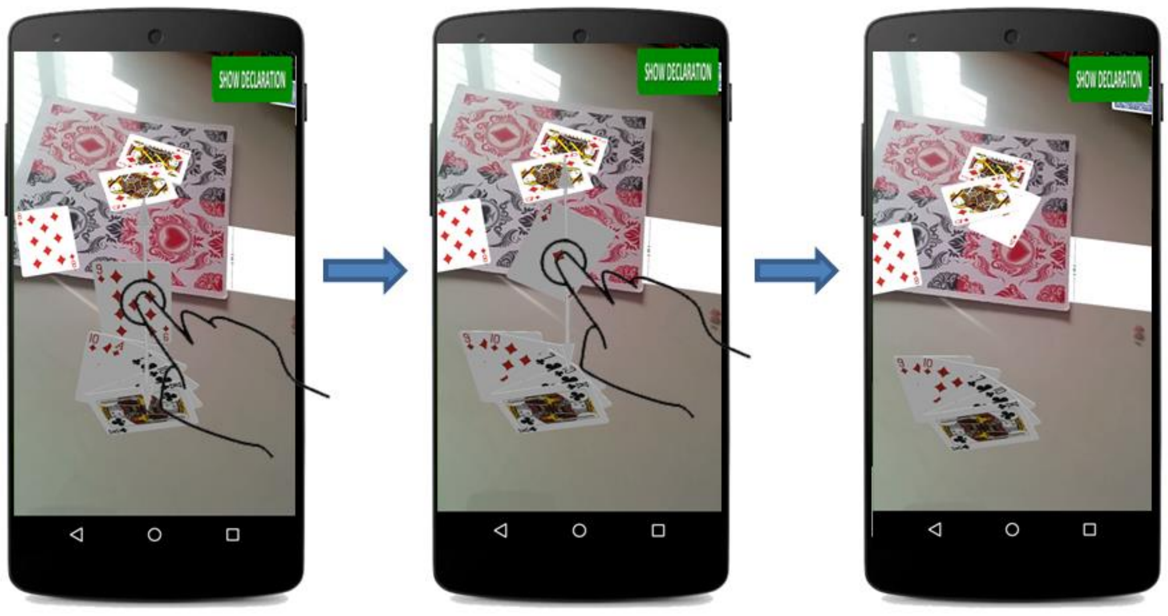


Figure 4.32: Incorrect card selected (1st screen) is placed back to its initial position and the correct card is selected (2nd screen) and thrown (3rd screen)

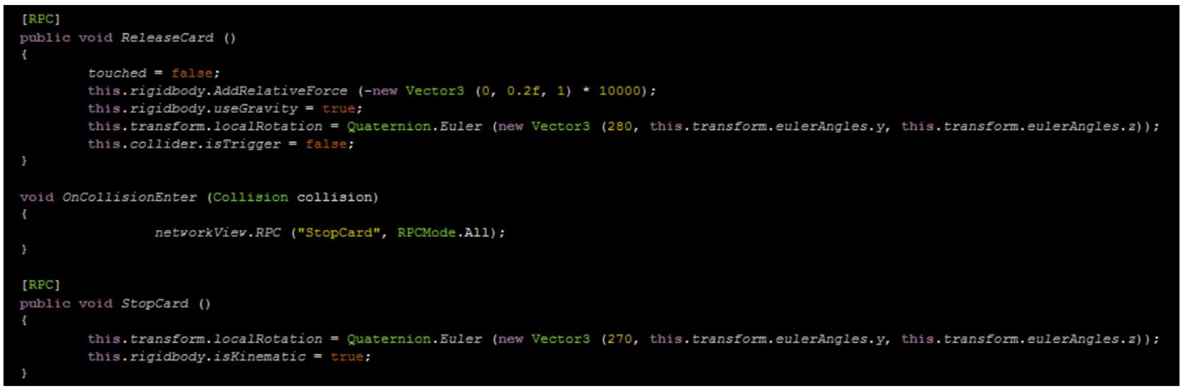
4.7.4 Game Physics

The physics, inside a game, play an important role in order to make the gameplay as realistic as possible. In this project the physics must be implemented very carefully so the main objective, which is to simulate realistically the traditional game, to be achieved.

Unity offers a very practical and user-friendly physics engine where developers can easily apply and modify the physics properties of an object. Initially, all the interacting gameobjects which are the floor and the cards, are applied a box collider component. For the floor, as stated previously, a big invisible box is created to simulate the collision of the cards to the table. Furthermore, for the 32 cards, a rigidbody component is attached to them because the cards have mass and need to obey the physics law. Both these gameobjects are also applied with a rubber material in order to eliminate as much as possible the friction between the cards and the floor and avoid the disappearance of any card from the gameplay area.

Next, the appropriate force needs to be applied on the throwing card so it will end up inside the gameplay area. The area that was decided to be the gameplay area is inside the image marker because is the only place that must be visible all the time for the game to be playable. A force is applied upon releasing the card multiplied by a threshold. This force is a negative vector3 (0, 0.2, 1) where the z component aims forward towards the table while the y component, together with the mass and gravity, directs the card downwardly.

After testing this force application, some issues were occurred. The first issue was that when the card was released by the player, it collided with the other holding cards and their positions were rearranged. To overcome this, the "IsTrigger" variable in the collider component was selected. This variable, when it is marked as true, allows objects with colliders to avoid any collision with other objects. Moreover, when the card is released the "IsTrigger" variable is changed to its false state. This modification makes a thrown card avoid collision with the other cards while allowing collision with the floor gameobject. The next issue was that, sometimes, when a card hit the floor, depending on the height that was released, it was flipped around to its back side so the value of the card was not visible. To avoid this behaviour a modification was made inside the OnCollisionEnter function that handles the collision of a gameobject. As soon as a card collides with the floor, its local rotation is changed so it will always be flipped to its value side and any errors to be avoided. Furthermore, when the card hits the floor the "IsKinematic" variable of the RigidBody component is set true. This variable when it is set true, stops any kind of force applied to an object, making it static. Figure 4.33 shows the final script of throwing a card towards the floor.



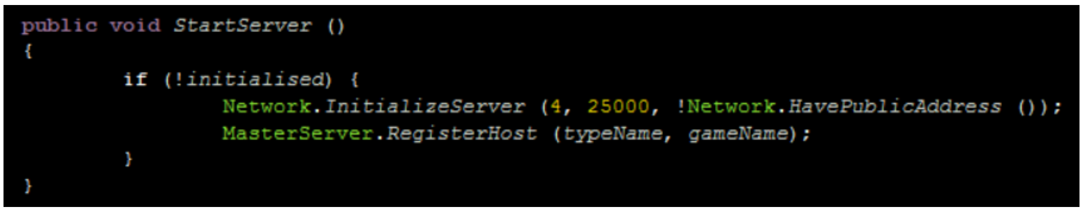
```
[RPC]
public void ReleaseCard ()
{
        touched = false;
        this.rigidbody.AddRelativeForce (-new Vector3 (0, 0.2f, 1) * 10000);
        this.rigidbody.useGravity = true;
        this.transform.localRotation = Quaternion.Euler (new Vector3 (280, this.transform.eulerAngles.y, this.transform.eulerAngles.z));
        this.collider.isTrigger = false;
}

void OnCollisionEnter (Collision collision)
{
                networkView.RPC ("StopCard", RPCMode.All);
}

[RPC]
public void StopCard ()
{
        this.transform.localRotation = Quaternion.Euler (new Vector3 (270, this.transform.eulerAngles.y, this.transform.eulerAngles.z));
        this.rigidbody.isKinematic = true;
}
```

Figure 4.33: Example code of ReleaseCard and StopCard function which is called when a collision occurs

4.8 Network Implementation

After implementing the AR technology to the prototype, all the appropriate functionality to play the game with the correct rules and interaction techniques, is set. The only piece that is missing in order for the game to be used by multiple users is the development of the networking. In the context of this thesis, Unity's Masterserver was used but in the future development the server will be moved to a computer with the correct capabilities to provide better potentialities for the users.

First of all, the game must be initialised into the Server so the players can join it. When the first player selects the "Create Game" button on the first screen the server is initialised (Figure 4.34), allowing four connections and listening port is set to 25000, so the game is registered into the MasterServer.



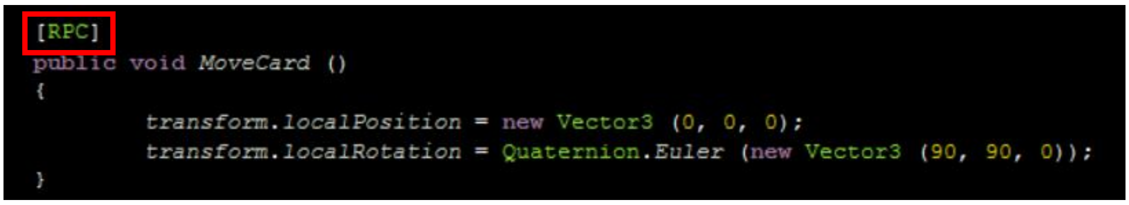
```
public void StartServer ()
{
        if (!initialised) {
                Network.InitializeServer (4, 25000, !Network.HavePublicAddress ());
                MasterServer.RegisterHost (typeName, gameName);
        }
}
```

Figure 4.34: Example code for initialising the Server

When the dealer created the server all the other three players must join this server by clicking "Join Game" button. When the players click the button the RefreshHostList() function requests a host list from the MasterServer. Due to the fact that this request may take several seconds to find the host list, a coroutine waits four seconds to ensure that the host list was obtained. When the four seconds are passed the JoinServer function connects players to the first host data of that host list. In this way, a correct client – server communication is achieved.

Continuing, the next functionality that needs to be implemented is the data sharing between the four players. Unity offers a NetworkView component for data sharing over the network. This NetworkView component must be applied to all the gameobjects that need to be synchronised across the network and for this project it is applied to all the 32 cards and also to the empty gameobject that contains all the scripts that handle in-play data. With this setup, any change that is made by the dealer (server) is synchronised to all the connected players.

For example, when the dealer throws a card, the card's gameobject position is shared to the other three players so everyone can see the movement of that card. When it comes to the other three players (clients) turn to throw a card, the card is not shared over the network so a modification must be made. The NetworkView component allows two kinds of network communication: State Synchronization and Remote Procedure Calls. In order to allow clients share their data the Remote Procedure Calls (RPC) are used. RPC allows user to call a function on the server so it can be synchronised over the network. To allow functions to be shared, the RPC tag must be applied over the specific function (Figure 4.35) inside the scripts.



```
[RPC]
public void MoveCard ()
{
        transform.localPosition = new Vector3 (0, 0, 0);
        transform.localRotation = Quaternion.Euler (new Vector3 (90, 90, 0));
}
```

Figure 4.35: RPC tag placed on top of the function that should be synchronised across network

The RPC tag is applied to all the functions that handle the position transformation of the cards and also to all the functions that change any values of the cards such as their power or player values such as the player turn.

4.9 Classes – Scripts

Besides the graphics, a dynamic coding must be developed to link all these graphics with actions and variables in order for the game to operate correctly. Some specific code snippets were presented in previous chapters and in this section all the script classes, available inside the game, will be analysed. Unity3D offers a variety of programming languages but for this project the chosen language is C Sharp. For C Sharp language there is a strong documentation online while it is supported by the official Unity forum with many example codes. The scripts that were developed are discussed below:

ShuffleDeal (RandomNum): This script is responsible for shuffling the 32 cards and dealing them to the four players. Moreover, while dealing the cards to each of the four players, it sorts them based on their holding power. The highest card of each suit is placed on the left side and the lowest to the right side. Furthermore, it places the four suits to an order of Hearts, Clubs, Diamonds and Spades so that the red and black cards to be separated. If a suit is not available in player's cards another arrangement is made so it will still follows the Red, Black pattern. The script is placed into the Gameplay gameobject.

Bidding (ShuffleandDeckTest): The specific script, placed into the Gameplay gameobject, contains all the functionality that handles the bidding round by the dealer. All the GUI textures that were placed inside the Hierarchy are added into this script, to initialise them as buttons. Moreover, this script calls the ShuffleDeal script to shuffle the deck and get the cards dealt to each player and place them into the correct positions. Finally, it saves all the information for the bidding round into variables that will be accessed by other scripts for counting the score.

Player (MouseDrag): This script is allocated to the ARCamera gameobject and handles the user interaction inside the main gameplay. Here, the procedure for selecting, throwing and declaring the correct cards is calculated.

Rules: Rules script is the most important script because it handles all the rules of such a game. The script contains all the functions that calculate the winner of each round, the final winner of the game and writing the score onto the paper gameobject. Moreover, all the GUI textures for adding the declarations, by the dealer, are added and initialised properly.

InPlayData: This script handles all the variables needed for the main gameplay. It contains all the variables and arrays that are used in each round such the CardOnTable array. It is responsible for initialising all the variables to their correct states after a round is finished such as the first played card variable.

NetworkManager: Inside this script, all the functionality for starting and connecting to the server is placed. Furthermore, the buttons for creating and joining a game are initialised correctly.

CardValues: This script is placed into each of the 32 cards and contains the values of each card independently. Besides the values, the script is responsible for every movement and force applied to a card such as the dragging before throwing a card. Furthermore, because it is allocated to each card it manages the collision of the cards when they hit on the table.

4.10 Game installation

All the applications that are developed with Android OS are installed into a device with the use of an APK file. This APK is a compressed file that compiles all the appropriate files such as the graphics, libraries and coding scripts into a mobile device. Figure 4.36 shows the steps for exporting the APK file from Unity game engine.

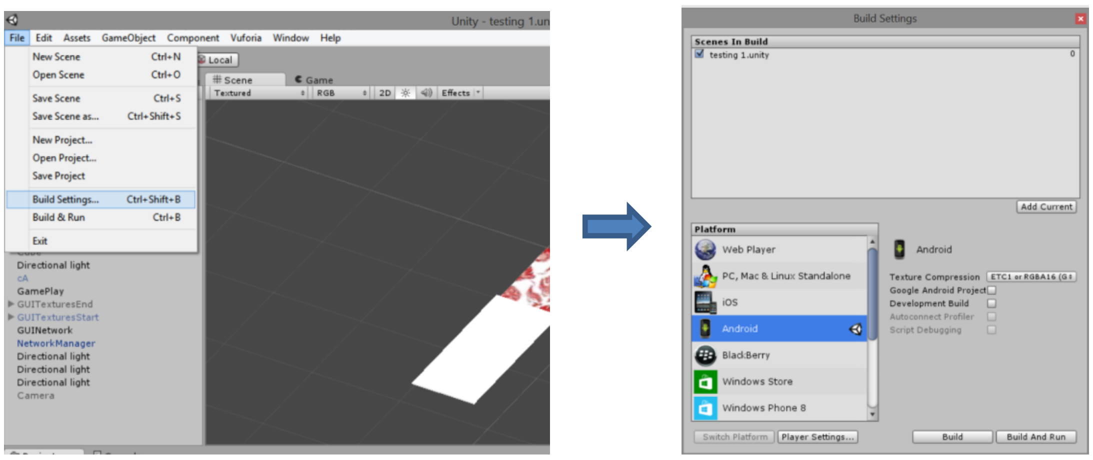
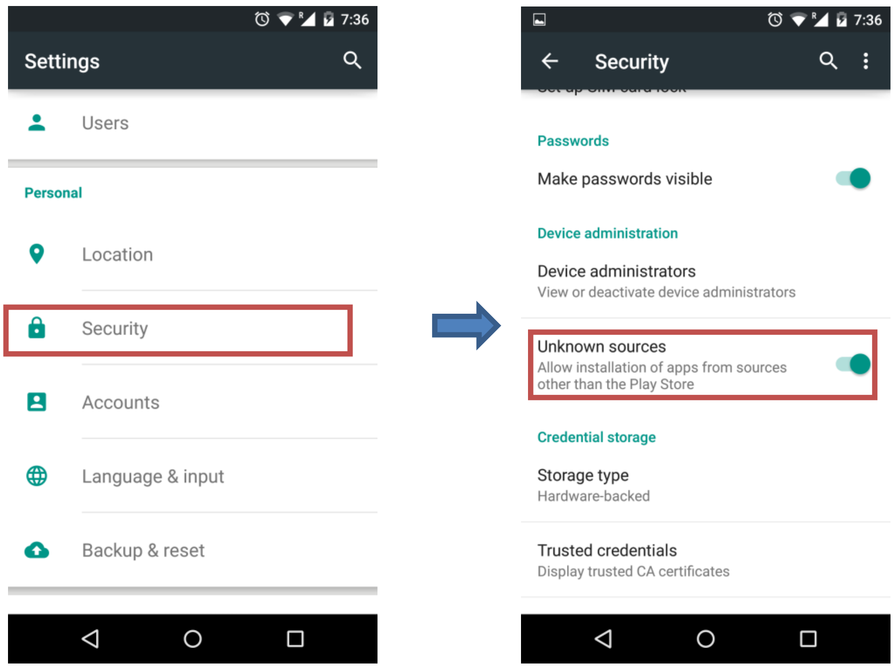


Figure 4.36: Exporting the APK file from Unity3D

In order to install the file, users need to move it into their device. This can be simply done by connecting the device to the PC using a USB cable and copy the file into the internal memory of the device. Because this file it will be manually installed by the user and not installed by the device's app store some steps need to followed (Figure 4.37) in order to authorise this installation.



Passwords

Make passwords visible

Device administration

Device administrators
View or deactivate device administrators

Unknown sources
Allow installation of apps from sources other than the Play Store

Credential storage

Storage type
Hardware-backed

Trusted credentials
Display trusted CA certificates

Figure 4.37: Enabling installation from unknown sources (Android 5.0+)

Besides this procedure, the file can also be shared via various sharing applications such Dropbox. The application can be placed inside a dropbox folder and when it is uploaded a download link can be shared to other users.

After placing or downloading the file inside the device's memory the user, with the use of a file manager application, can locate the file and install it.

4.11 Final prototype description

This section presents the final prototype running on four different android devices while analysing each part of the game. Table 4.4 shows the four devices that were used with their specifications.

Table 4.4: Specifications of the devices used

Device	Operating System	Camera's Resolution	Device's Screen
Nexus 5	Android 4.4.4	8 MP	4.95 inches
Samsung Galaxy s3 mini	Android 4.1	5 MP	4.0 inches

Samsung Galaxy R	Android 4.0	5 MP	4.2 inches
Samsung Galaxy Ace2	Android 2.3	5 MP	3.8 inches

Moreover for the setup, four mobile stands were used for easier gameplay as shown in Figure 4.38. The AR marker was printed and placed on top of a thick piece of cardboard in order to be stable. Finally, all four devices are connected to the same network and the game can start.

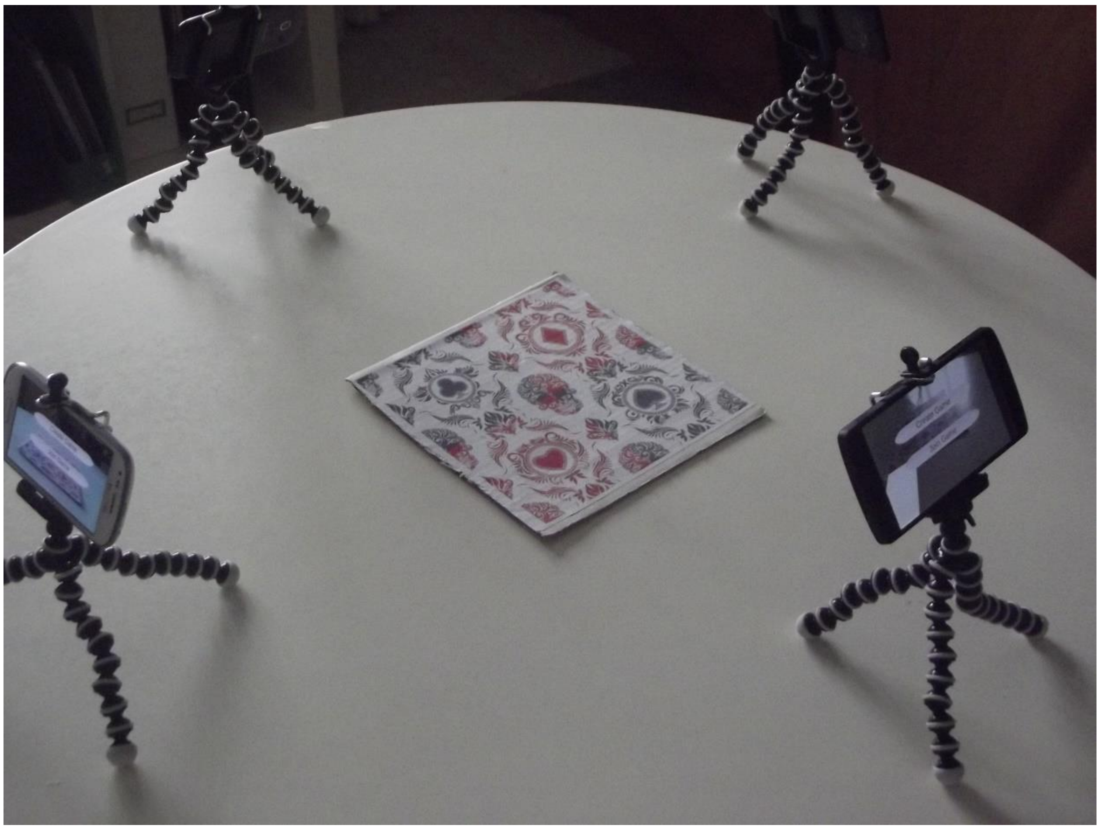


Figure 4.38: Setup used for running the game on the four devices

4.11.1 Starting the Game

The game is installed and launched by the four devices. The first screen appears to the devices as shown in Figure 4.39.

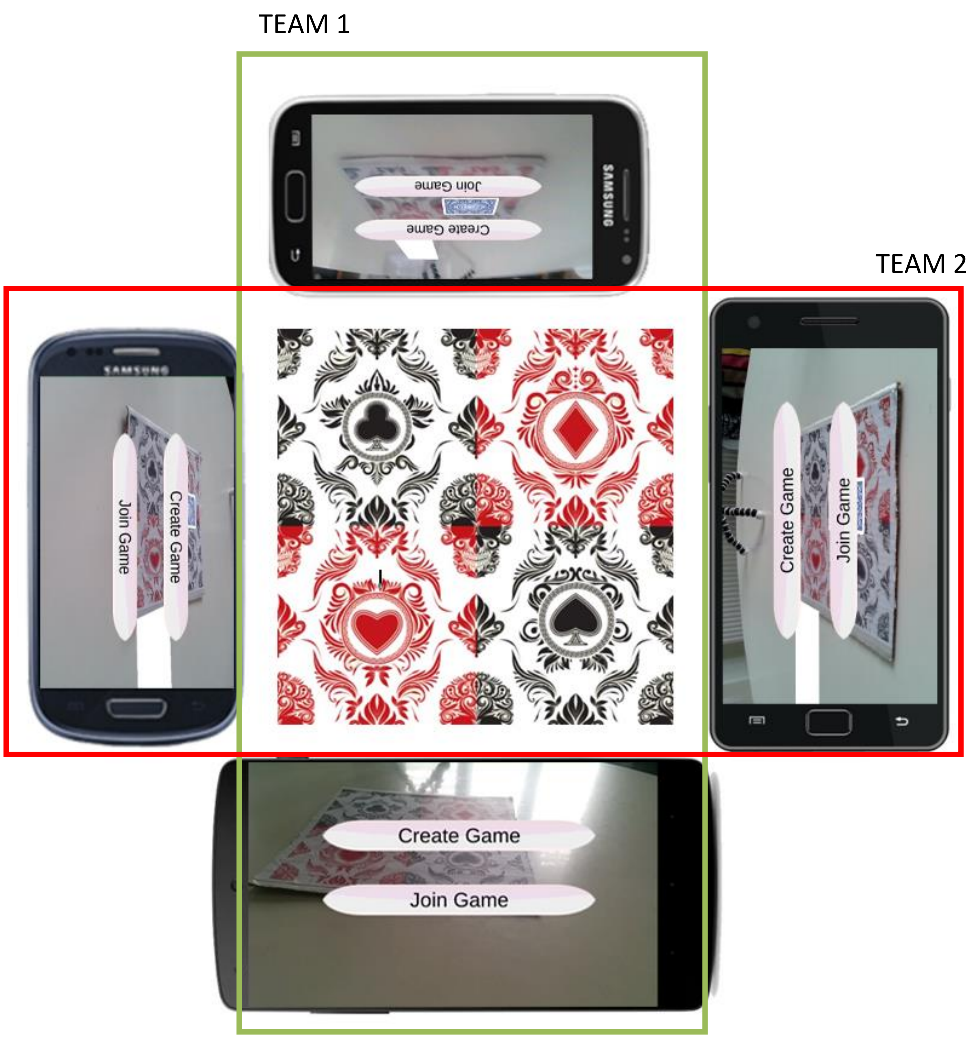


Figure 4.39: GUI displayed on launching the game

The dealer was decided to be the player with the Nexus 5 so it clicks the “Create Game” button and waits for the other three players to join it. In the dealer’s screen appears the “Shuffle” button but is not clickable until all the players are connected (Figure 4.40).

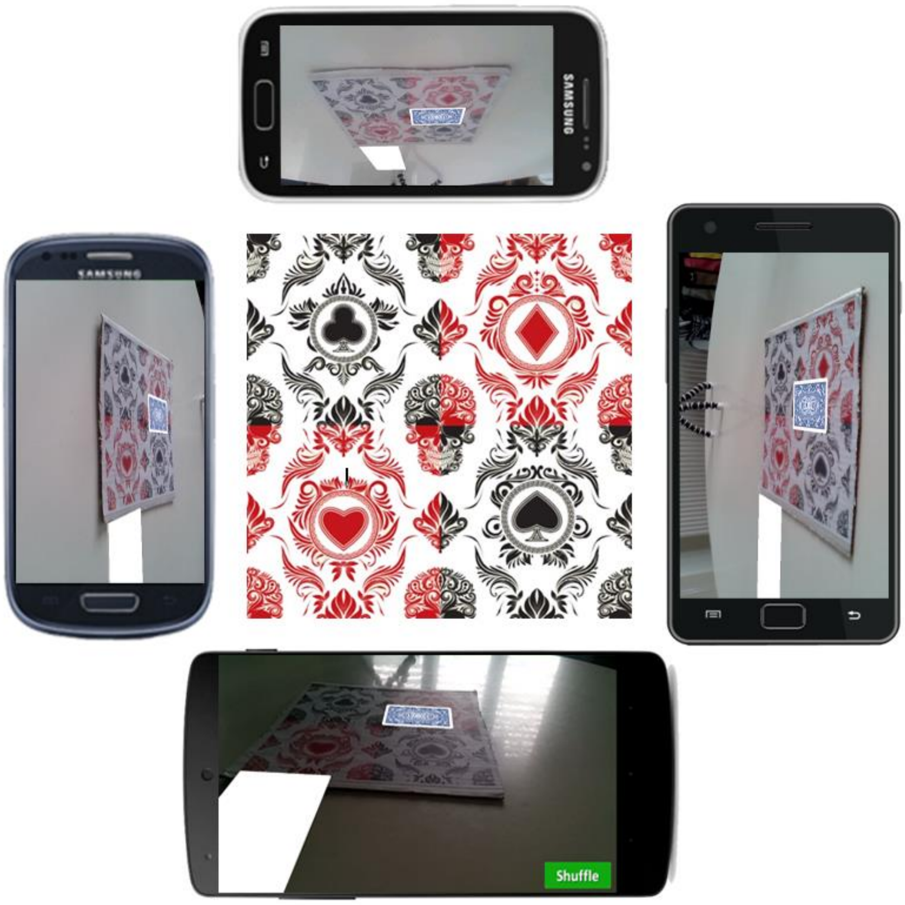


Figure 4.40: Dealer created a game and all three players joined it

The other three players accordingly click the "Join Game" button to connect to the created game. Four seconds after clicking the button, the players are all joined together and waiting the dealer to shuffle and deal the cards. The dealer clicks the "Shuffle" button and the cards are distributed to the players (Figure 4.41) and the bidding round begins.

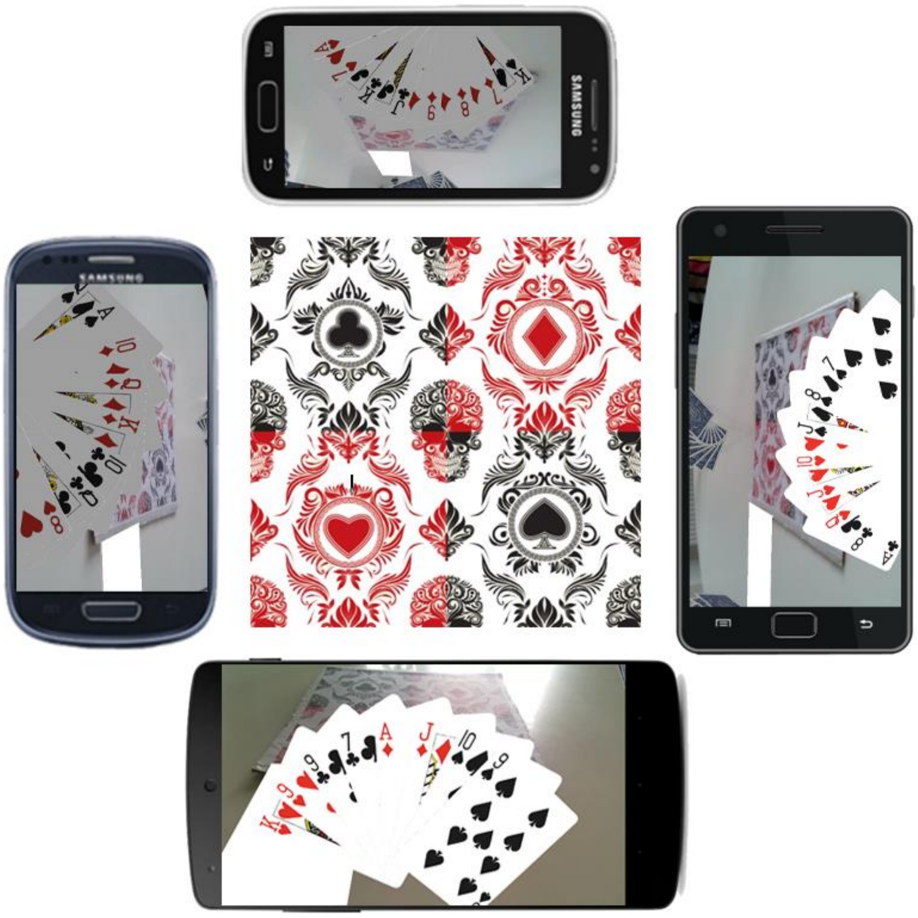


Figure 4.41: All 32 cards dealt to the four players

4.11.2 Bidding Round

The players check their cards and start bidding. The first player who bids is the one right to the dealer and on each round the first bidding turn goes to the next player accordingly. Each player either bids or passes until three passes occur (Figure 4.42).

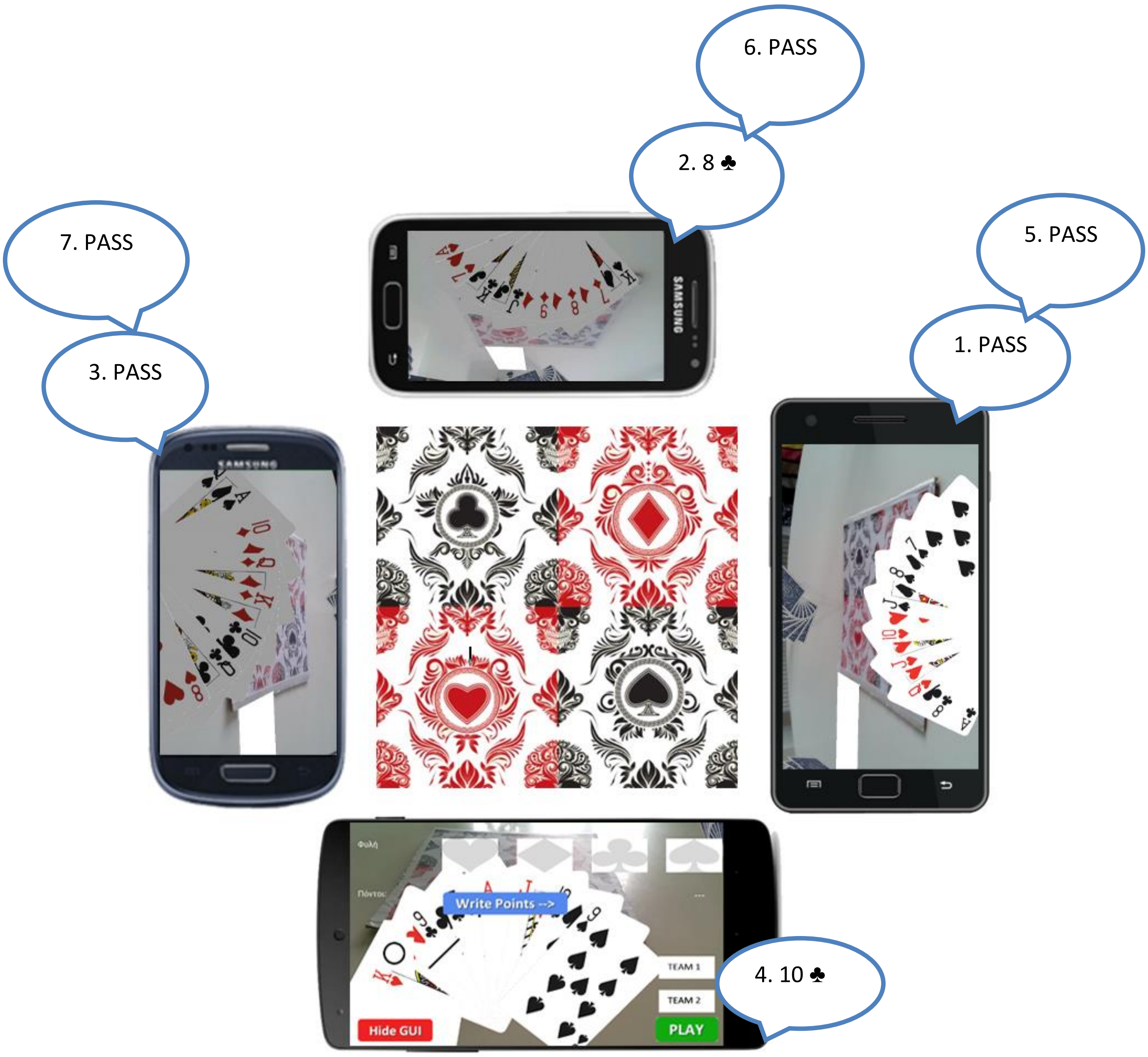


Figure 4.42: Bidding round; the first player to speak (Player 2) passes, Player 3 bids 8 of spades, Player 4 passes and dealer (Player 1) adds two points to his teammate. Then the next three players pass so the bidding round is finished

When the three passes occurred, the dealer declares on his/ her screen the suit, the bid and the team who bid last. Figure 4.43 shows the declared bidding by the dealer who clicks the "Play" button to start the main gameplay.

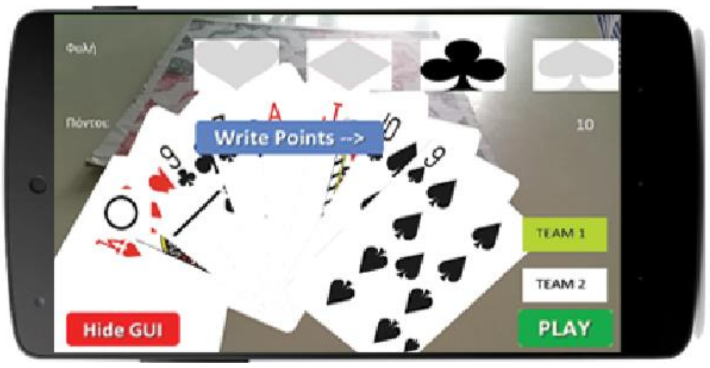


Figure 4.43: Dealer declares the final bid (10 of spades) on his team (Team 1)

4.11.3 Main Gameplay

After the dealer pressed the “PLAY” button, the four players’ screen is changed to main screen where only the “Show Declarations” button appears. The gameplay as already mentioned is based on a turn-by-turn way so each player waits until his/ her turn. At the first round, any player who has a declaration waits his/ her turn and informs the other players that he/ she has a declaration. In order to win the extra points of that declaration, the player must remember to show the declaration to the other three players on the second round or else the points are lost. Figure 4.44 and Figure 4.45 below show the interaction of the players and how the declarations are shown respectively.

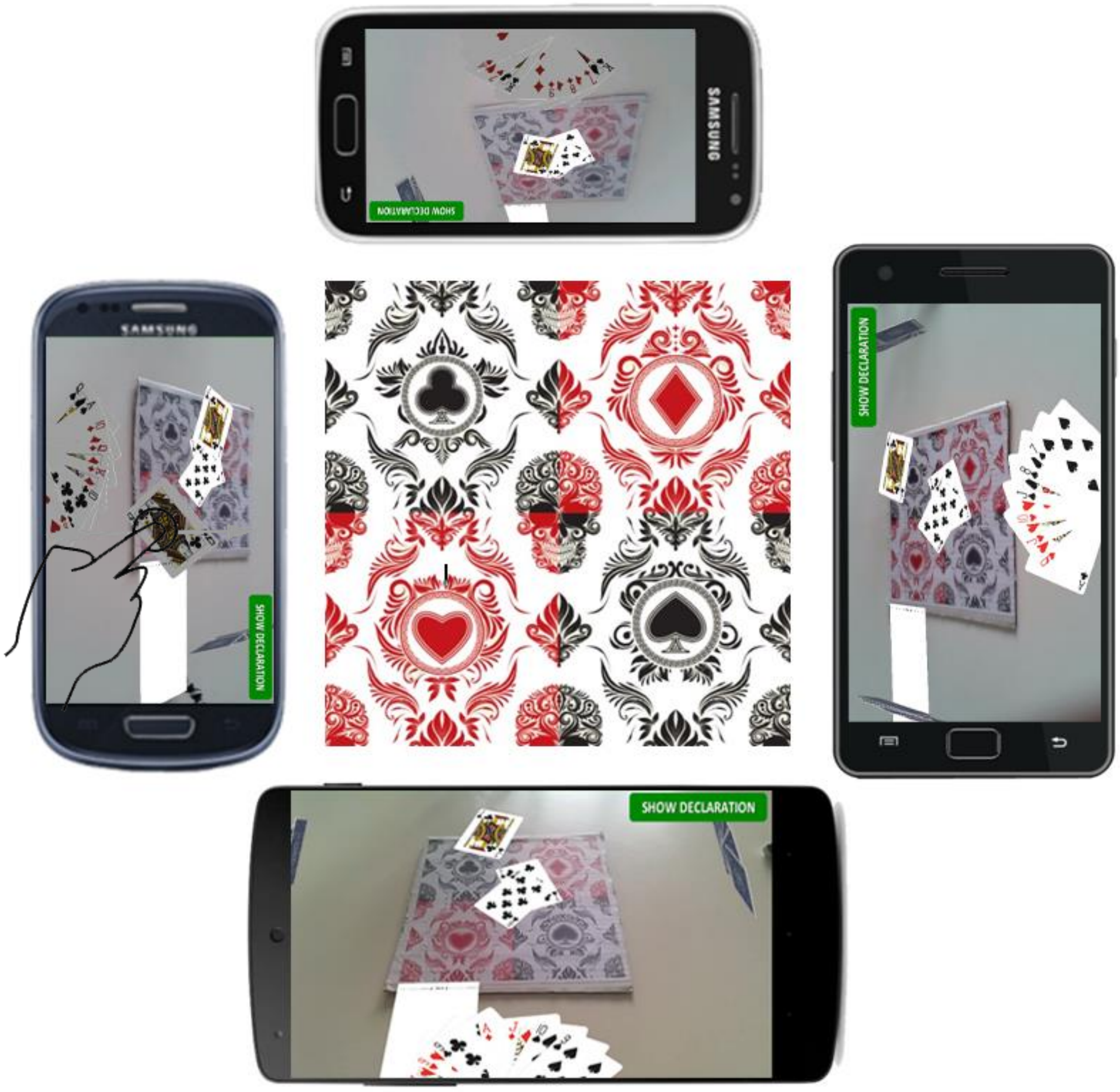


Figure 4.44: Player's 4 turn to throw his card on the table, where the other three thrown cards are displayed

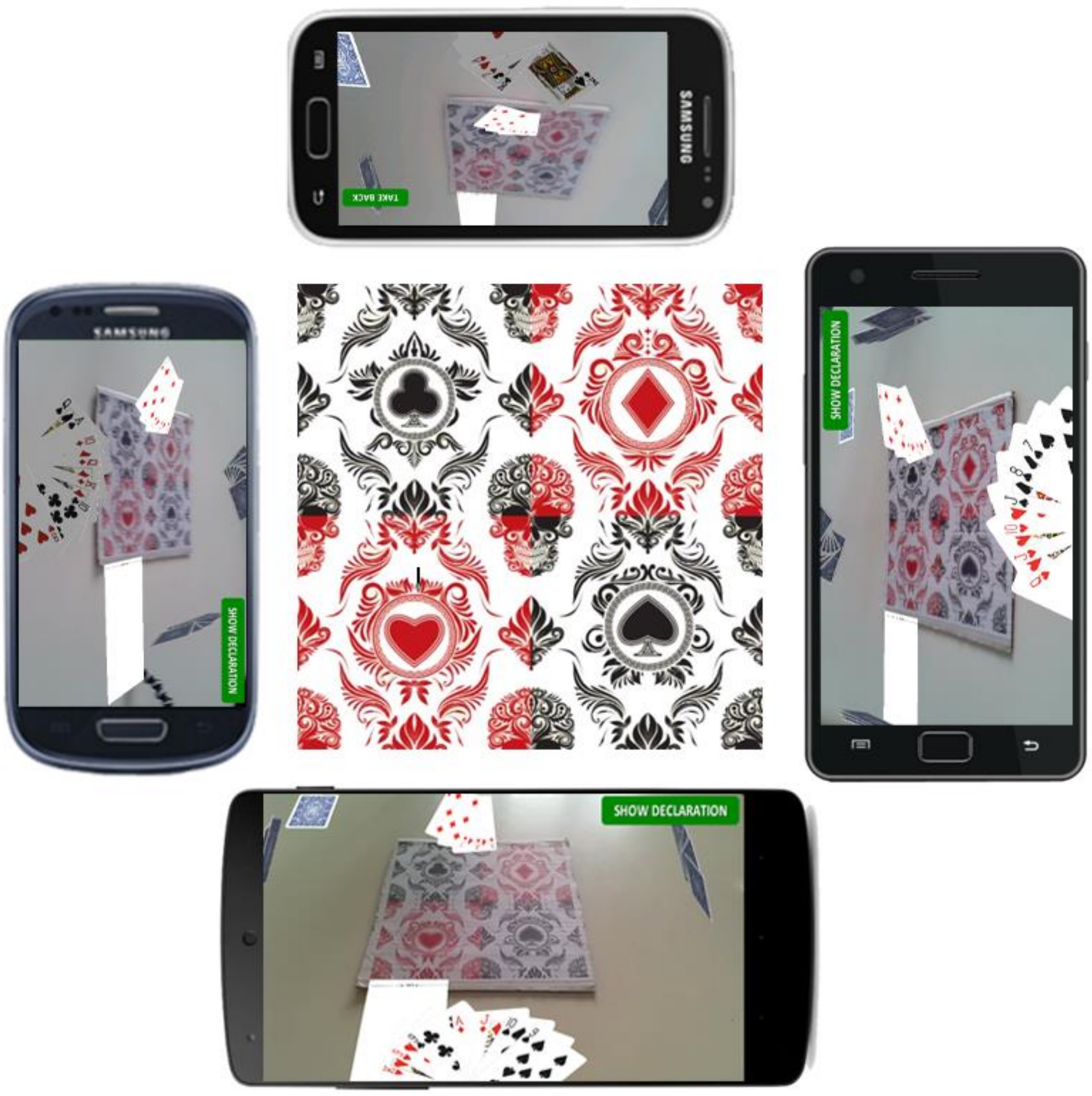


Figure 4.45: Player 3 shows his declaration which is displayed on all the other three players

4.11.4 Points

After the eight rounds are played, the dealer must add any extra points from any declarations shown in the main gameplay. When the points are added, the dealer clicks the “Count” button and the score is calculated and displayed on the paper. Then the dealer clicks the “Shuffle” button and a new game is started with the same procedure. The game is finished when one of the two teams reaches the maximum score that was declared before starting the game. Figure 4.46 shows how this process is done by the dealer and Figure 4.47 shows the score printed on the virtual paper.

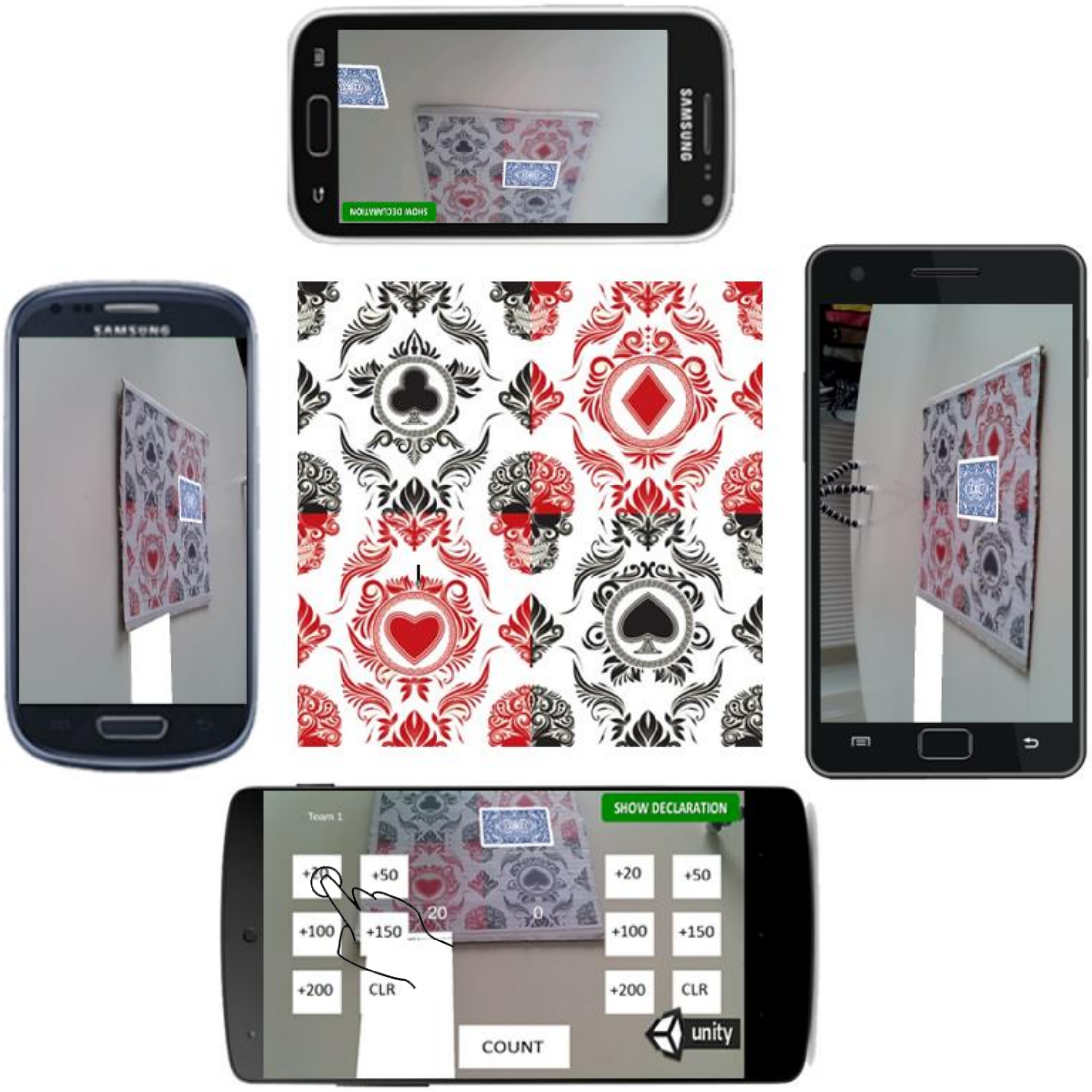


Figure 4.46: Counting points screen where the dealer adds 20 extra points on Team 1 because of the declaration shown by Player 3.

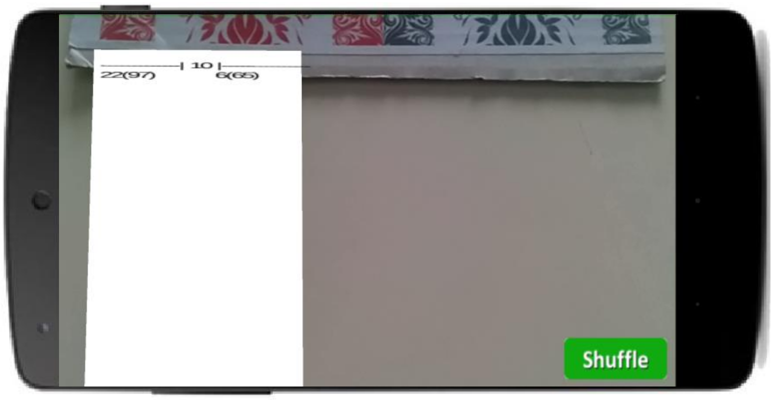


Figure 4.47: Score written on virtual paper. On the left side are Team's 1 score and on the right side Team's 2 score. On top is written the final bid of that round

4.12 Conclusion

In this chapter the developing techniques and software used for implementing the game were analysed. At first, an initial prototype was developed so that all the game's rules would be implemented. After reviewing and correcting the initial prototype, it was modified so that the AR technology could be added. Finally, the networking was implemented so that the game's data could be synchronised to all four players.

5 Evaluation

5.1 Introduction

After finishing the final prototype of the game, an experimental evaluation was carried out with real users in order to check the functionality of the AR game and evaluate if such a game can replace the traditional one.

5.2 Participants

For this experimental evaluation, four groups were created with four players each group. Each player should have their Android smartphone or else they were provided one. All sixteen players were advanced players of the traditional game as well as smartphone users, aged 20 - 26. Only two of the sixteen players knew about AR technology before using the application.

5.3 Procedure and Setup

For the experiment's setup, each group was provided with the AR marker used in section 4.10 which was placed in the middle of the table were each group was situated. Each group created two teams and sit accordingly around the table. Moreover, each group was asked to join the local network while deciding who was going to be the dealer in order for that person to select and create the new game where the other three players could join. The final score was decided to be at 101 for all four groups which took around 30 minutes, from the first to the last game, for each group. The main guidance given to all players was to try and play the game as they would have played the traditional game.

5.4 Methodology

5.4.1 Questionnaire

In the end of each game the sixteen players were asked to fill a questionnaire (ANNEX A3). This questionnaire was divided into three sections: User Interface, Gameplay and General Comments. The purpose of this questionnaire was to evaluate the functionality of the whole game and get feedback for all positive and negative aspects of the player's experience.

5.4.2 Observation

During each of the four games observation notes were taken on how the players were playing the game. This observation helped to compare the behaviour of the players playing the AR game with the traditional way of playing the game.

5.5 Results and Discussion

5.5.1 Questionnaire results

Table 5.1: Questions' mean score result

No	Question	Mean Score (Best = 5, Worst = 1)
1	Was it easy to Create/ Join a game	3.87
2	Was it understandable to declare the game (dealer)	4.25
3	Was it easy to declare the game (dealer)	2.5

4	Was it understandable to show a declaration	4.12
5	Was it easy to show a declaration	4
6	Was it understandable to write the declarations (dealer)	4.25
7	Was it easy to write the declarations (dealer)	4.5
8	Was the GUI user friendly in general	4
9	Was it easy to understand game controls	3.94
10	Was it easy to throw a card	3.8
11	Was it easy to scan and track the marker	3.5
12	Were the objects realistic	4.5
13	Was the position of the cards convenient	3.75
14	Performance of camera and mobile device	3.37
15	Were you aware of other players moves	4.31
16	Was it fun	4.81
17	Was it easy to use	4.12
18	Was it interesting	4.68
19	Was it realistic	4
20	Could it replace the traditional game	2.06

Table 5.1 shows the results of the questionnaire by all players, giving the mean score of each question. The results showed that in general the game was fun to play and easy to use giving a positive feedback for the functionality of the game. Below all individual answers are displayed into charts and analysed.

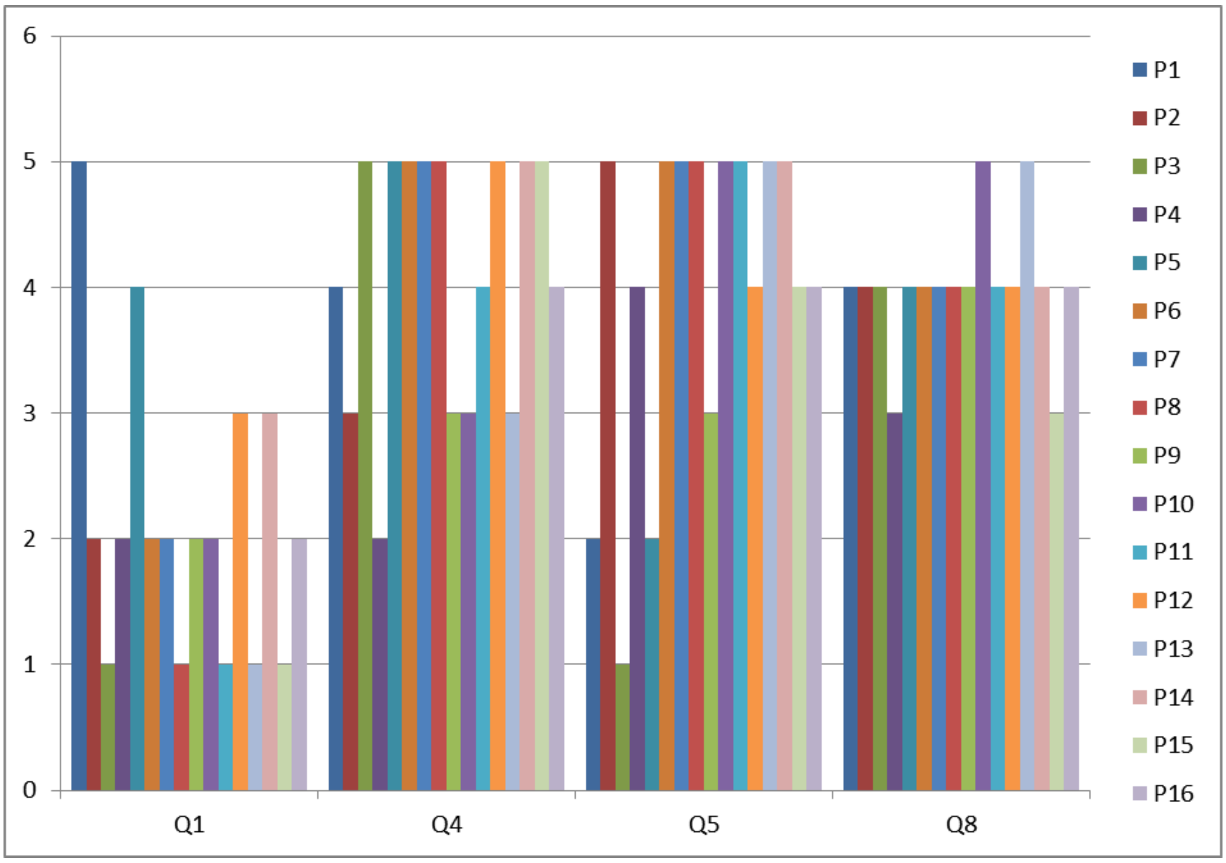


Figure 5.1: Individual results of Questions 1, 4, 5 and 8

Starting with the graphical part of the game, the chart above shows the individual results for questions 1, 4, 5 and 8 (Figure 5.1). The first question got a mean score of 3.87 which is a little above average because two players stated that was a little difficult to connect or create a game while the majority found it easy. Both those players that were dealers stated that although they created a game, they did not have any feedback whether the other players had joined his game. Moving on questions 4 and 5, almost all players stated that was understandable the way of showing their declarations in the main game but when it came to show their declarations some of them found it a bit hard. The reason was that they were not sure whether selecting a card would throw it to the table or not so, some guidelines were provided that by clicking a card would flip it in order to be visible to the other players. Question 8 results show that all players, despite those difficulties, found the overall GUI user-friendly.

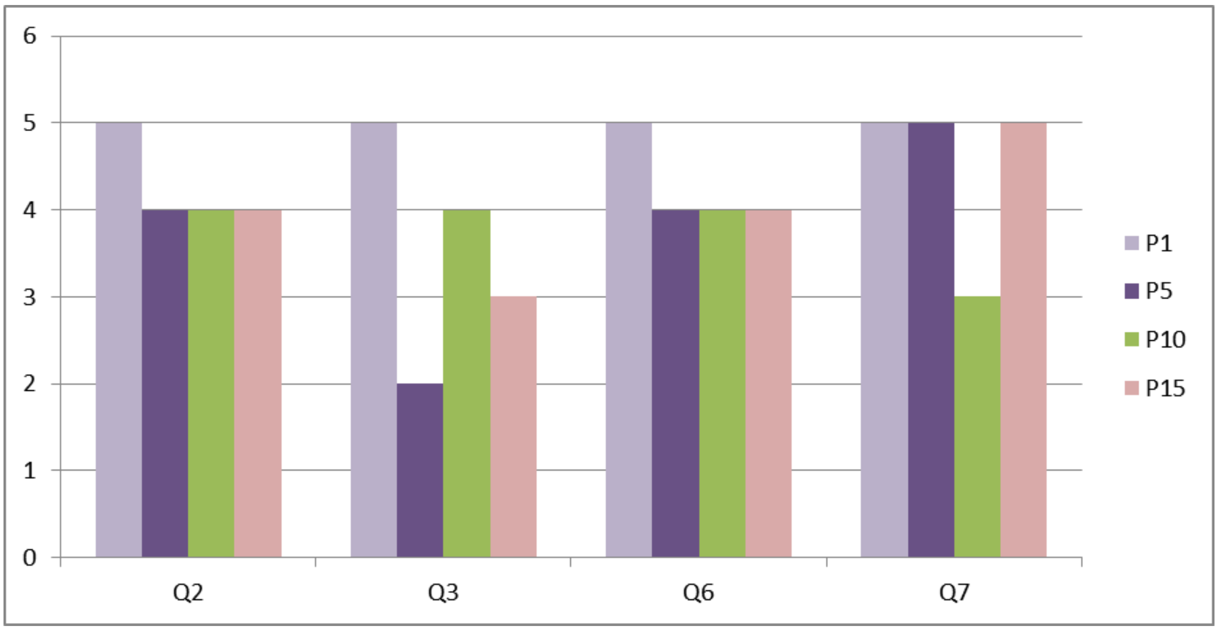


Figure 5.2: Individual results of Questions 2, 3, 6 and 7

Furthermore, questions 2, 3, 6 and 7 concerned only the four dealers as they were about the bidding and counting screens that are only available on dealers' devices (Figure 5.2). Starting with the bidding screen (Q2, Q3) all four dealers stated that the interface was understandable, but when they actually declared a game the opinions varied. Player's 5 score was low because as he stated the procedure of opening the device's keyboard to declare the bid was really slow. The reason was that his device had a slow processor so opening the keyboard inside a heavy game took a bit longer than usual. Moreover, player 5 and player 15 said that there should be a way for editing the final bid, inside the main gameplay, in case of a mistake. The counting points screen results show that all four players understood how the buttons worked and found it easy when using it. Although, three of the dealers stated that there should be a background texture behind the number of the points because the white font colour that is used was, sometimes, invisible.

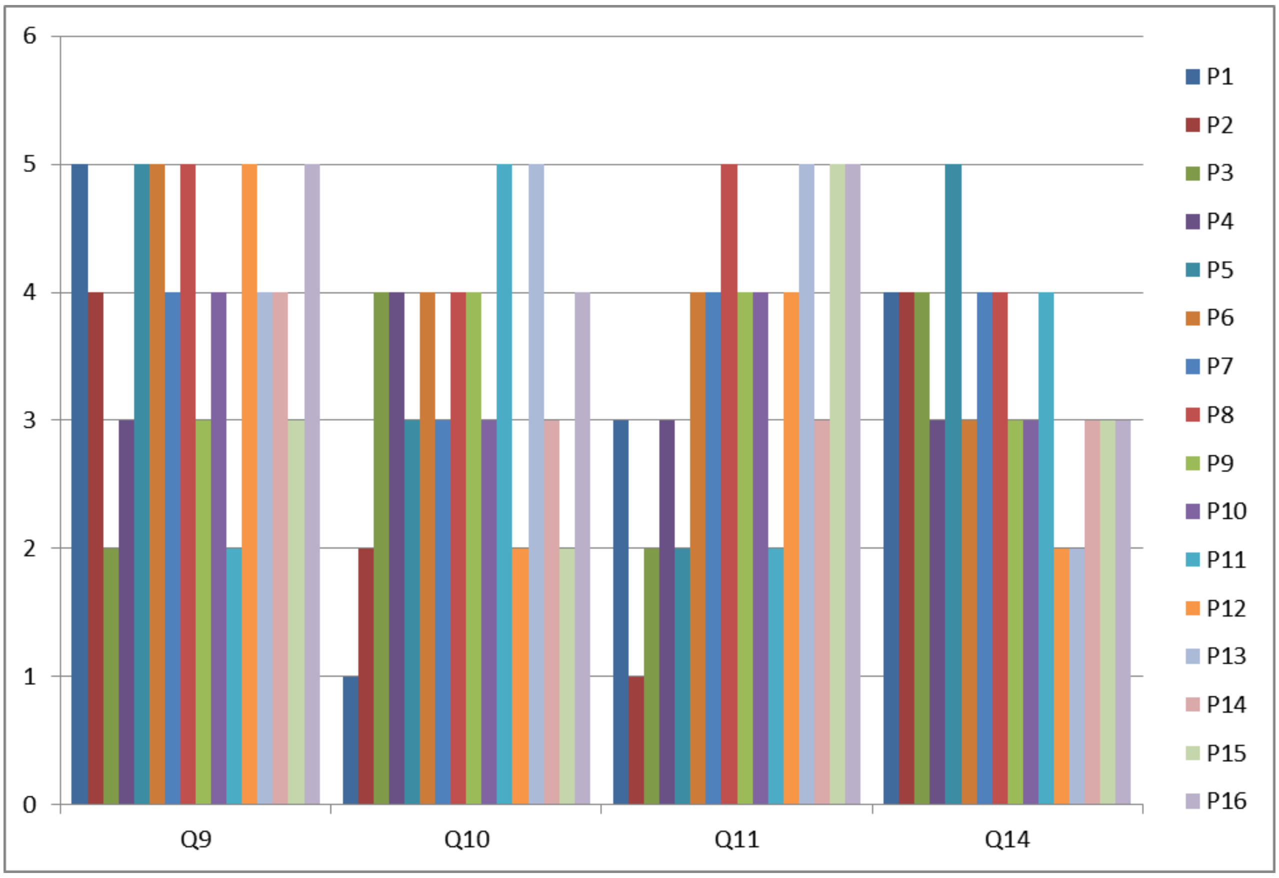


Figure 5.3: Individual results of Questions 9, 10, 11 and 14

Continuing to the gameplay functionality, questions 9, 10, 11 and 14 (Figure 5.3) were about the general interaction of the players with the game. Question 9 results show that the majority of the players understood the game controls while five of them needed some extra information. When the players came to throw their cards on their turn (Question 10) almost half of them found it easy where the other half had some problems. The main problem stated was that the distance between the cards was small leading them to select a wrong card. Moreover, players stated that they really liked the "drag-n-throw" way for throwing their cards but the threshold distance (section 4.7.4) should get smaller for quicker interaction. Moving on question 11 the opinions differed on how easy was to track the marker. This is also related to question 14 because the tracking depends heavily on the performance of the device. This is the reason for that variation in the rankings because users' devices differ from one another in respect of processor power. On low processor devices the tracking and the gameplay in general was running slower from high processor devices.

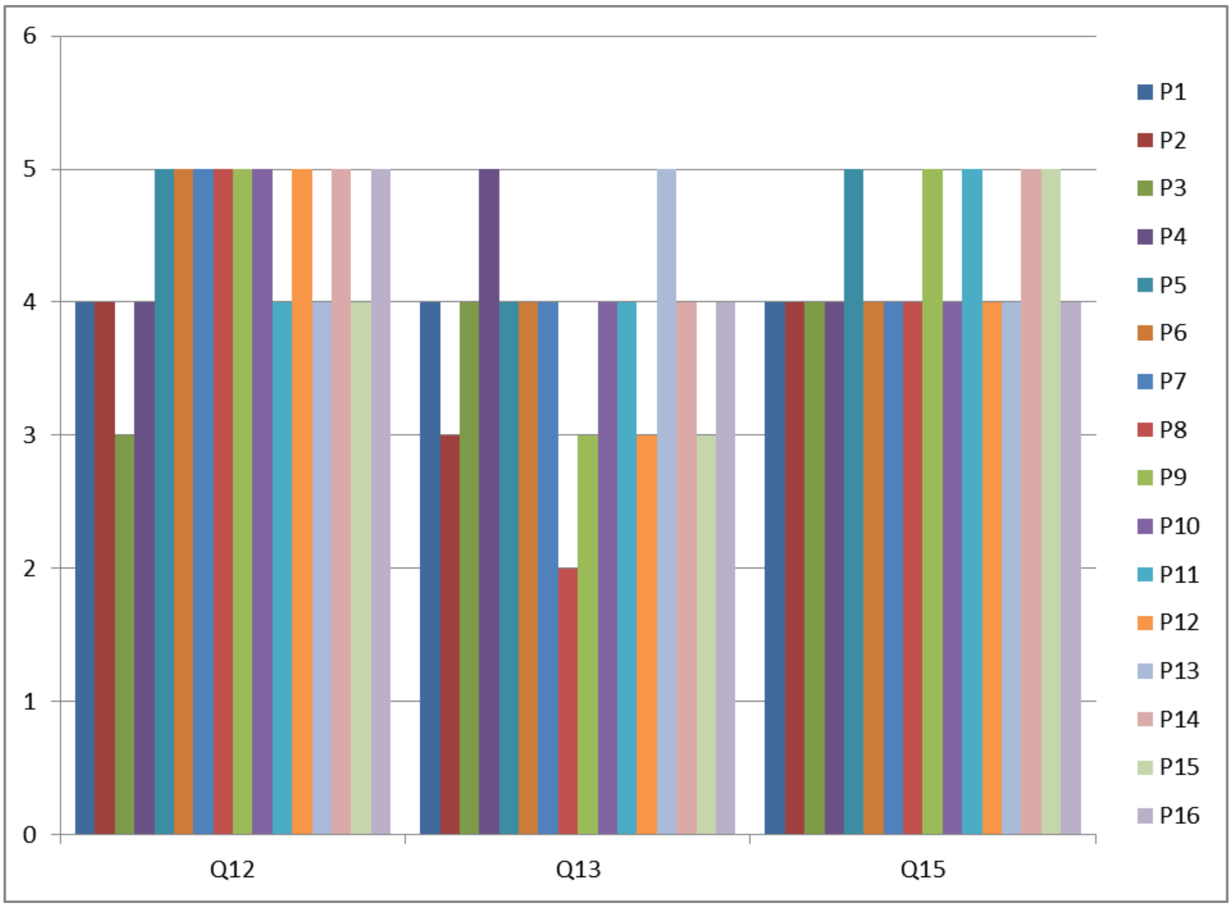


Figure 5.4: Individual results of Questions 12, 13, and 15

Next, the three questions above (Figure 5.4) are about the objects inside the game and the general setup of these objects. Almost all sixteen players rated above average that the elements inside the game were realistic which means that the objective of creating a game with realistic virtual elements is achieved. Moreover, the positioning of the cards was rated above average from the majority of the players while only one rated it below average. Player 8 stated that the position of the cards of each player was very close each other and recommended to increase the distance to avoid any cheat peeking from the players. Moving on question 15, all players stated that they were aware of the other players' moves. Some of the players recommended that the force applied to the cards should be more controllable because sometimes cards were thrown a bit outside of the marker.

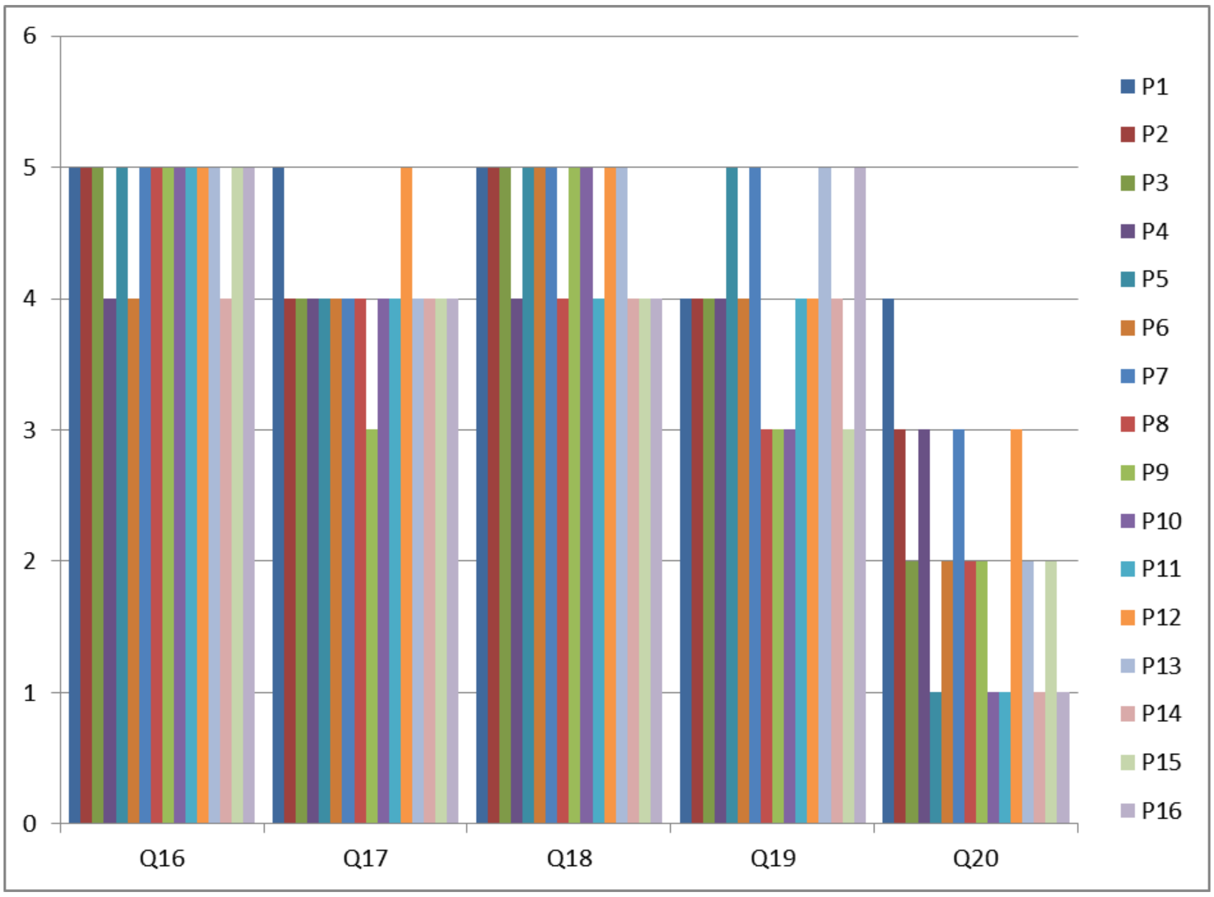


Figure 5.5: Individual results of Questions 16, 17, 18, 19 and 20

Finally, the last five questions (Figure 5.5) are concerned on the general opinion about the overall game. The majority of the players rated above average questions 16, 17, 18 and 19 showing that the game is easy and fun to play, the AR implementation is interesting and the gameplay with the elements was realistic and very close to the traditional game. Although most of the players stated that the AR game cannot yet replace the traditional game.

In the comments section of the questionnaire, players addressed some missing gameplay functionality in respect of the traditional game. The first issue that was stated by the second group was the absence of a "re-shuffle" button in the bidding round. This reshuffle occurs when all four players in the bidding round, passed without opening a bid so the dealer must reshuffle the deck. The next issue stated from seven players was the lack of the option for a player who has the strongest cards remaining to end the round. For example, in the traditional game, when a player knows that his/ her cards are the winning cards for all the following rounds he/ she can state it to the other players and finish the specific game earlier. Despite those issues, it was also commented that the automate way of dealing the cards and counting the points satisfied them because as stated from nine of them "the manually dealing and counting are really boring tasks". Moreover, twelve players stated that they really liked the "drag-n-throw" way for throwing their cards. Finally, fourteen players commented that they were impressed by the AR technology and that they are really looking forward in future expansion of the game.

5.5.2 Observation Results

This section describes all the observation notes taken while watching the users playing the AR game. At first, when the players first tracked the marker and the virtual cards were displayed on their screen, they were very impressed and started exploring the whole area of the marker to view their cards from all perspectives. Moreover, the players seemed to have fun playing the game. Sometimes players even looked at the marker placed on the table to view the thrown cards which shows that the immersion level was high tricking the players that they had actually threw the cards on the physical table. Though, this high immersion made the players to forget the main guideline that they were given stated before after playing the first rounds. The reason was that they were mostly focused on their device's screen thereby reducing the social part of the game. This resulted in some cases to forget whose turn was to play causing minor confusion and slowing the gameplay.

Furthermore, some players had difficulties tracking the marker because their device's camera was not powerful enough and even the minor movement could result to the loss of tracking. The device played a big role in gameplay performance. Devices with slow processor had issues on displaying the virtual objects causing low frame rate and making the interaction very slow. Furthermore, many users mentioned that their devices were heat up and the battery was drained very fast.

5.6 Conclusion

In general, the evaluation has shown that users found the game very interesting and the AR technology very impressive. Although, the issues stated, as well as the results on the 20th question on the questionnaire, show that such a game cannot be totally replaced by an AR game but there is a positive feedback that with further appropriate development of this technology may someday replace traditional games and processes in general.

6 Conclusion & Future Work

Augmented Reality technology in the last two decades made huge development steps. Along with the rapid evolution of mobile devices from heavy devices into portable “smart” phones with high hardware capabilities, AR made its entrance into our everyday lives. This migration offered new opportunities, as discussed in Chapter 2, in many areas such as marketing and education. Moreover, AR proposed new ways of interaction in gaming as well as allowing players to share not only the virtual space but also their physical surroundings increasing the immersion level. Though, almost all AR games created augment unrealistic virtual data that do not blend with the physical space and also most of them do not allow multiple players.

In this research project an AR multi-user game was developed to explore the multiplayer aspect of AR gaming. Furthermore, the traditional card game of Pilotta (Belote) was chosen to be built so the realism of the virtual objects to be succeeded. Augmenting virtual data such as deck cards increases the realism in such a virtual world because cards can easily be created and applied a realistic texture to look exactly as the actual physical cards. Achieving this realism, AR games could possibly replace the traditional way of playing such games.

Before the implementation, a research was conducted to examine how card game players behave while playing such games as well as to discover the positives and negatives aspects of these games in order for the requirements to be exported as discussed in Chapter 3. Furthermore, when designing such an AR game several techniques and software can be used. For this project, the marker-based tracking technique was used for augmenting the objects with Unity3D game engine and Vuforia plugin.

In Chapter 4, the implementation techniques are analysed in detail including the implementation of all the rules applied in such game, the merge of the AR technology and finally the implementation of the networking. Implementing such a game proved to be a complex procedure that needs to be evaluated at each step. Continuous evaluation minimised possible errors in respect of the game rules resulting to a compact game in respect of graphics and gameplay mechanisms.

The evaluation of the final prototype was conducted with real users in order for the functionality and the overall user experience to be checked. The results have shown that in respect of the functionality of the game, users were very satisfied and impressed by this new, for them, AR technology experience. On the other hand, users stated that such an AR game cannot yet replace the traditional way of playing the card game. The reason for that was mostly based on the performance of the device. In respect of hardware performance, most smartphones were unable to perform AR operations smoothly, leading to low satisfactory level in respect of user's experience. AR applications perform a lot and heavy calculations that require the processor to operate in full speed consuming large amount of battery. Although, this problem can be outlined as a minor issue because processors and batteries used in smartphones are advancing every year. In general, the resulting game can be considered as a success into the AR gaming industry because it offers new ways for expanding AR games such as the multiplayer option and the realism aspect.

6.1 Future work

The future expansion of the game has already been started exploring new ways for expanding the portability of the marker-based tracking method. The marker-based method as discussed limits the users to carry and use a piece of paper in order to be able to play the game. Moreover a tester had stated that for someone else sitting around the table that does not have the game to see the virtual objects that are thrown to the table seems kind of funny.

That led to the thought of creating a more interactive marker rather than just a piece of paper. For that reason, a new application was designed that can be imported into a mobile device with big screen and can be used as a marker. More precisely, a new scene was created inside the Unity project that contained all the assets from the AR game scene. The only difference is that the ARCamera was replaced by a virtual camera positioned above the marker and its field of view covered the whole marker as well as a part of the virtual paper. Furthermore, the device running this application can also connect with the other four players. The application it was successfully tested on a tablet as shown in Figure 6.1. This allows players to track the marker from the tablet's screen while viewing all the cards that are thrown on top of it. This increases, not only the portability of the marker-based method, but also the user's experience with the game because it lets them view the cards on the table without changing their camera position.

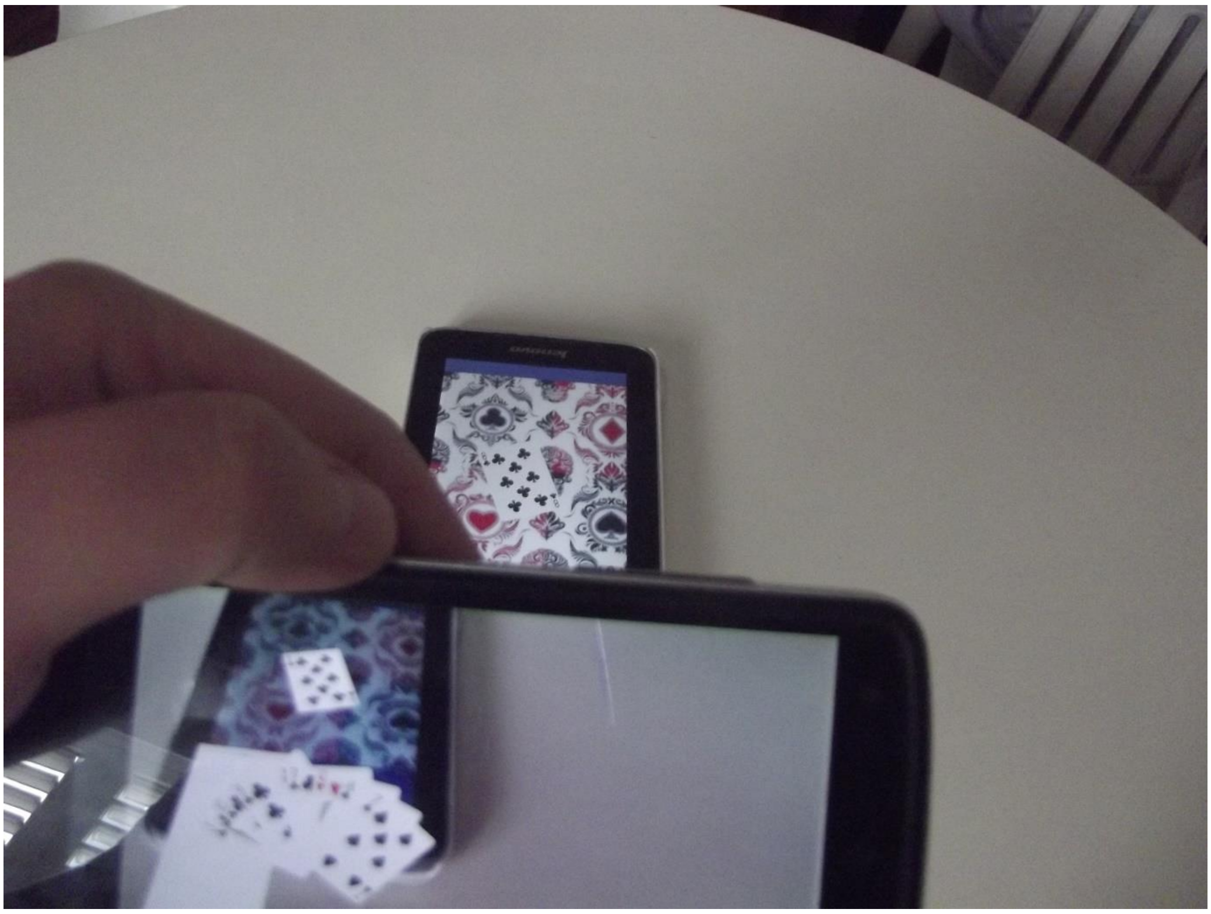


Figure 6.1: Marker displayed on tablet

Furthermore, another setup was created with the use of mobile projector. The specific application is running on a laptop using an Android emulator and a projector, which is connected on the laptop, projects it on the table surface (Figure 6.2). Such a setup can be used in the future, in places such as cafeterias or even at home where potential players can play the game carrying and using only their mobile devices.

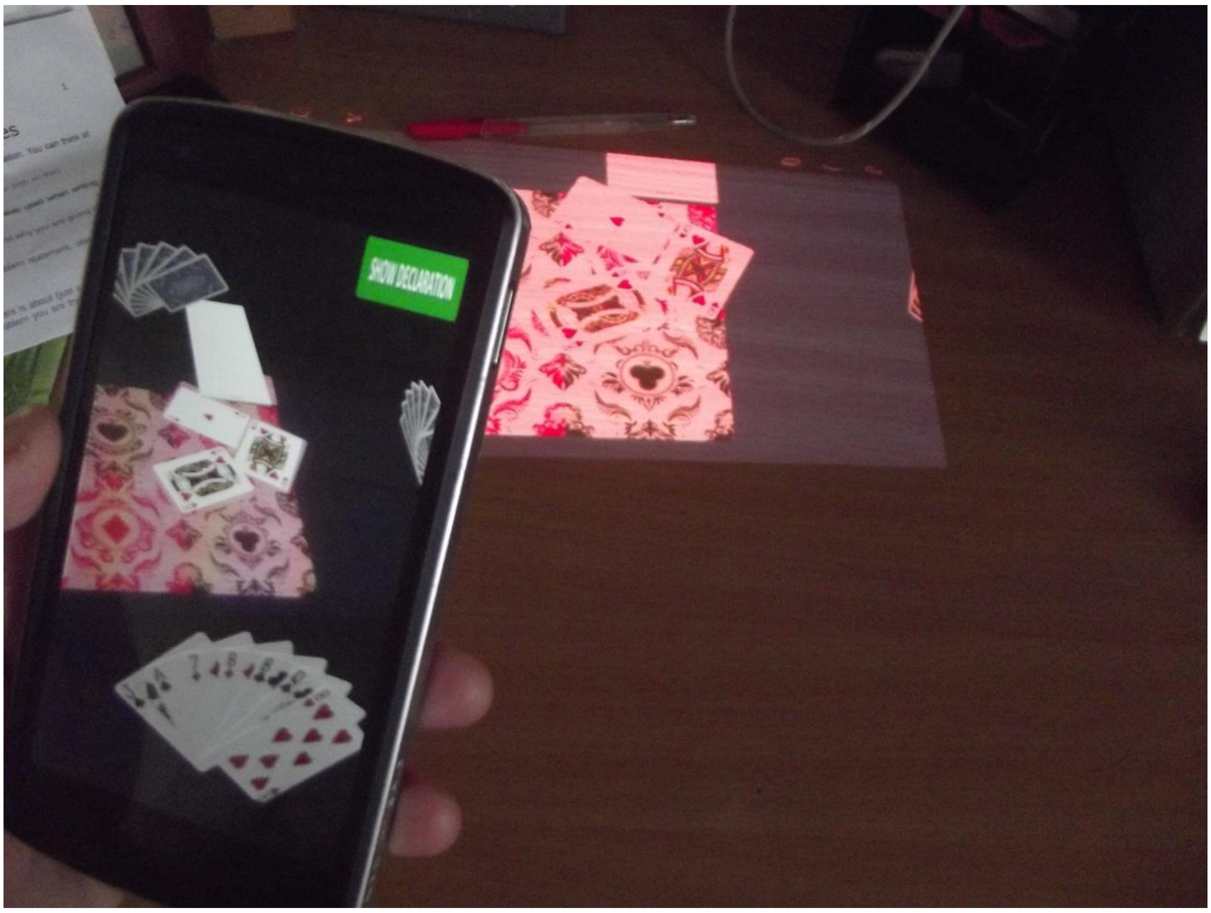


Figure 6.2: Marker projected on table

Besides that, future work on the game includes all the issues and improvements stated in the evaluation results. The most common issue was the confusion of whose turn was to play. This can be implemented using a text that is displayed on all four players indicating the turn. Furthermore users claimed that viewing the score was a bit difficult. For that reason, rather than having a paper object that displays the score, the score can be placed as a text on the players screen.

Finally, future expansions of the game include improvements of the existing interface and availability for other Operating Systems. Expanding the application to other Operating Systems such as IOS and Windows would make it more popular by increasing the number of users. Also, making the game available in multiple languages will make it accessible to users all around the world.

BIBLIOGRAPHY

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., & MacIntyre, B. (2001). Recent advances in augmented reality. Computer Graphics and Applications, IEEE, 21(6), 34-47.

Barakonyi, I., Weilguny, M., Psik, T., & Schmalstieg, D. (2005, June). MonkeyBridge: autonomous agents in augmented reality games. InProceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology (pp. 172-175). ACM.

Baricevic, D., Lee, C., Turk, M., Hollerer, T., & Bowman, D. A. (2012, November). A hand-held AR magic lens with user-perspective rendering. InMixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on (pp. 197-206). IEEE.

Billinghurst, M., Belcher, D., Gupta, A., & Kiyokawa, K. (2003). Communication behaviors in colocated collaborative AR interfaces. International Journal of Human-Computer Interaction, 16(3), 395-423.

Bulearca, M., & Tamarjan, D. (2010). Augmented Reality: A Sustainable Marketing Tool?. Global Business and Management Research: An International Journal, 2(2&3), 237-252.

Chang, Y. N., Koh, R. K. C., & Duh, H. L. (2011, October). Handheld AR games—A triarchic conceptual design framework. In Mixed and Augmented Reality-Arts, Media, and Humanities (ISMAR-AMH), 2011 IEEE International Symposium On (pp. 29-36). IEEE.

Chuptys, S., & De Coninck, J. Head Mounted Displays.

Craighead, J., Burke, J., & Murphy, R. (2007). Using the unity game engine to develop sarge: a case study. Computer, 4552, 366-372.

Desai, P. R., Desai, P. N., Ajmera, K. D., & Mehta, K. (2014). A Review Paper on Oculus Rift-A Virtual Reality Headset. arXiv preprint arXiv:1408.1173.

Diaz, M., Alencastre-Miranda, M., Munoz-Gomez, L., & Rudomin, I. (2006, November). Multi-user networked interactive augmented reality card game. InCyberworlds, 2006. CW'06. International Conference on (pp. 177-182). IEEE.

Drummond, T., High Speed Matching and Tracking. Retrieved from: http://www.qualcomm.com/media/documents/high-speed-matching-andtracking-slides

Feiner, S. (2002). Augmented Reality: A new way of seeing. Scientific American.

Google Cardboard. (2014, July 14). Retrieved from https://cardboard.withgoogle.com/

Henrysson, A., & Ollila, M. (2003, October). Augmented reality on smartphones. In 2nd Augmented reality Toolkit Workshop.

Huynh, D. N. T., Raveendran, K., Xu, Y., Spreen, K., & MacIntyre, B. (2009, August). Art of defense: a collaborative handheld augmented reality board game. In Proceedings of the 2009 ACM SIGGRAPH symposium on video games (pp. 135-142). ACM.

Kasahara, S., Heun, V., Lee, A. S., & Ishii, H. (2012, November). Second surface: multi-user spatial collaboration system based on augmented reality. InSIGGRAPH Asia 2012 Emerging Technologies (p. 20). ACM.

Kim, Y. G., & Kim, W. J. (2014). Implementation of Augmented Reality System for Smartphone Advertisements. International Journal of Multimedia & Ubiquitous Engineering, 9(2).

Lavender, T., & Gromala, D. (2012). Portable Presence: Can Mobile Games be Immersive Games?.

Lee, K. (2012). Augmented reality in education and training. TechTrends, 56(2), 13-21.

Mackay, W. E. (1998, May). Augmented reality: linking real and virtual worlds: a new paradigm for interacting with computers. In Proceedings of the working conference on Advanced visual interfaces (pp. 13-21). ACM.

Molla, E., & Lepetit, V. (2010, October). Augmented reality for board games. InMixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on (pp. 253-254). IEEE.

Nilsen, T., Linton, S., & Looser, J. (2004). Motivations for augmented reality gaming. Proceedings of FUSE, 4, 86-93.

Okada, H., & Arakawa, H. (2010). Augmented Reality Applied to Card Games.Augmented Reality, Soha Maad (ed.), 175-184.

Rohs, M. (2007). Marker-based embodied interaction for handheld augmented reality games. Journal of Virtual Reality and Broadcasting, 4(5), 1860-2037.

Rubino, D., Michaluk, K., Nickinson, P., & Ritchie, R. (2014). Can Mobile Gaming Kill The Consoles. Retrieved from http://www.androidcentral.com/talk-mobile/can-mobile-gaming-kill-consoles

Silva, R., Oliveira, J. C., & Giraldi, G. A. (2003). Introduction to augmented reality. National Laboratory for Scientific Computation, 11.

Stapleton, C., Hughes, C., & Moshell, M. (2002). Mixed Reality and the Interactive Imagination: Adding the art to the science and technology of Mixed Reality for training, education and entertainment. In First Swedish-American Workshop.

Szalavári, Z., Eckstein, E., & Gervautz, M. (1998, November). Collaborative gaming in augmented reality. In Proceedings of the ACM symposium on Virtual reality software and technology (pp. 195-204). ACM.

Szalavári, Z., & Gervautz, M. (1997, May). Using the personal interaction panel for 3d interaction. In proceedings of the Conference on Latest Results in Information Technology (p. 36).

Tan, C. T., & Soh, D. (2010). Augmented reality games: A review. Proceedings of Gameon-Arabia, Eurosis.

THE LEADING GLOBAL GAME INDUSTRY SOFTWARE. (2014). Retrieved from http://unity3d.com/public-relations

Thomas, B. H. (2012). A survey of visual, mixed, and augmented reality gaming. Computers in Entertainment (CIE), 9(3), 3.

Uchiyama, H., & Marchand, E. (2012, February). Object detection and pose tracking for augmented reality: Recent approaches. In 18th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV).

Välkkynen, P., Boyer, A., Urhemaa, T., & Nieminen, R. (2011). Mobile augmented reality for retail environments. In Proceedings of Workshop on Mobile Interaction in Retail Environments in conjunction with MobileHCI.

Vestola, M. A Comparison of Nine Basic Techniques for Requirements Prioritization. Helsinki University of Technology.

Wagner, D., & Schmalstieg, D. (2003, October). First steps towards handheld augmented reality. In 2012 16th International Symposium on Wearable Computers (pp. 127-127). IEEE Computer Society.

Wagner, D., & Schmalstieg, D. (2007). Design Aspects of Handheld Augmented Reality Games.

Wagner, D., Pintaric, T., Ledermann, F., & Schmalstieg, D. (2005). Towards massively multi-user augmented reality on handheld devices (pp. 208-219). Springer Berlin Heidelberg.

Xu, Y., Gandy, M., Deen, S., Schrank, B., Spreen, K., Gorbsky, M., ... & MacIntyre, B. (2008, December). BragFish: exploring physical and social interaction in co-located handheld augmented reality games. In Proceedings of the 2008 international conference on advances in computer entertainment technology (pp. 276-283). ACM.

ANNEXES

A1. Card Games Questionnaire

1. Gender *
Mark only one oval.

- Male
- Female

2. Age *
Mark only one oval.

- 15 - 20
- 20 - 30
- 30+

3. What card games do you like to play? *
(Please choose your favorite)
Mark only one oval.

- Biriba
- Pilotta
- Poker
- Kun Kan
- Prefa
- Other:

4. How often do you play? *
Mark only one oval.

- Every day
- 2-3 times a week
- 4-5 times a week
- 1-2 times a month
- 3-4 times a month

5. Where do you usually play? *
Mark only one oval.

- Home
- Cafe
- Other:

6. What do you like in the game? *
Tick all that apply.

- Teamwork
- Competition
- Chatting with friends
- Other:

7. What DON'T you like in the game? *

Tick all that apply.

- Always Bringing the deck
- Bringing always pen and paper
- Counting the score
- Deal Cards (μοίρασμα)
- Other:

8. Have you ever played that game electronically? *
Mark only one oval.

- Yes Skip to question 9.
- No Skip to question 13.

9. if yes, What did you like? *
----

10 .if yes, What didn't you like? *
----

11. Do you use your mobile phone while playing? *
Tick all that apply.

- Yes, for texting
- Yes, for playing other games
- Yes, for browsing
- No
- Other:

12. Will you be interested in a mobile game form? *
Mark only one oval.

- Yes
- No

13. Do you use your mobile phone while playing? *
Tick all that apply.

- Yes, for texting
- Yes, for playing other games
- Yes, for browsing
- No
- Other:

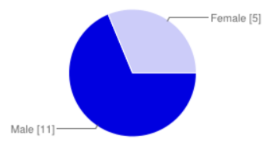
14. Will you be interested in a mobile game form? *
Mark only one oval.

- Yes
- No

A2. Card Games Questionnaire (Results)

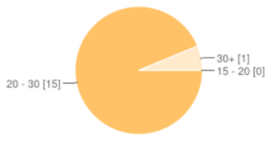
1)

Gender



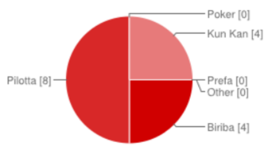
Male	**11**	65%
Female	**5**	29%

Age



15 - 20	**0**	0%
20 - 30	**15**	88%
30+	**1**	6%

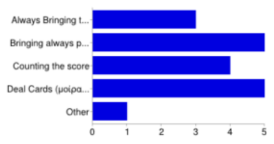
What card games do you like to play?



Biriba	**4**	24%
Pilotta	**8**	47%
Poker	**0**	0%
Kun Kan	**4**	24%
Prefa	**0**	0%
Other	**0**	0%

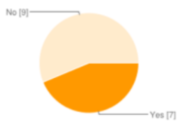
3)

What DON'T you like in the game?



Always Bringing the deck	**3**	18%
Bringing always pen and paper	**5**	29%
Counting the score	**4**	24%
Deal Cards (μοίρασμα)	**5**	29%
Other	**1**	6%

Have you ever played that game electronically?



Yes	**7**	41%
No	**9**	53%

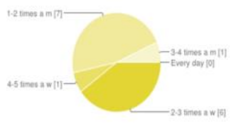
Page 2

if yes, What did you like?

that i can play pilotta online!!
meet new players and how they think
not dealing
the fact that i could play with friends in different countries
the whole game
the randomness

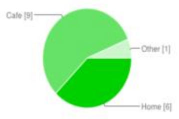
2)

How often do you play?



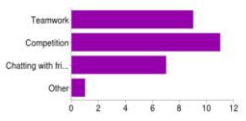
Every day	**0**	0%
2-3 times a week	**6**	35%
4-5 times a week	**1**	6%
1-2 times a month	**7**	41%
3-4 times a month	**1**	6%

Where do you usually play?



Home	**6**	35%
Cafe	**9**	53%
Other	**1**	6%

What do you like in the game?



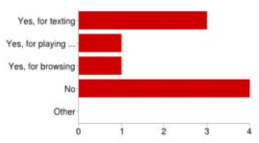
Teamwork	**9**	53%
Competition	**11**	65%
Chatting with friends	**7**	41%
Other	**1**	6%

4)

if yes, What didn't you like?

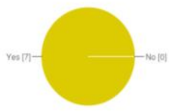
you dont have direct touch with the other players , and for that kind of games you neet the body language of the players
nothing
If I played with bots it wasnt competitive enough.
server problems
the moves of cpu were predictable sometimes
the gui

Do you use your mobile phone while playing?



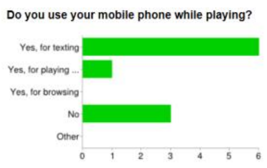
Yes, for texting	**3**	18%
Yes, for playing other games	**1**	6%
Yes, for browsing	**1**	6%
No	**4**	24%
Other	**0**	0%

Will you be interested in a mobile game form?

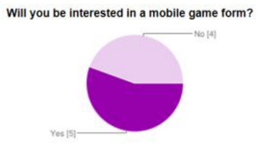


Yes	**7**	41%
No	**0**	0%

5)



Yes, for texting	6	35%
Yes, for playing other games	1	6%
Yes, for browsing	0	0%
No	3	18%
Other	0	0%



Yes	5	29%
No	4	24%

A3. Final Questionnaire

Παρακαλώ βάλτε σε κύκλο το σωστό:

1) Φύλο:

Αρσενικό Θυληκό

2) Ηλικία:

15-20 21-30 30+

3) Πόση εμπειρία έχετε με Mobile Games;

Καθόλου 1 2 3 4 5 **Πάρα Πολλή**

4) Πόση εμπειρία έχετε με την Πιλόττα;

Καθόλου 1 2 3 4 5 **Πάρα Πολλή**

5) Πόση εμπειρία είχατε με την Augmented Reality τεχνολογια πριν από το παιχνίδι;

Καθόλου 1 2 3 4 5 **Πάρα Πολλή**

6) Τι κινητό και τι λειτουργικό σύστημα χρησιμοποιήθηκε;

…………………………………………………………………………………………….

ΠΕΡΙΒΑΛΛΟΝ ΧΡΗΣΤΗ (USER INTERFACE)

7) Πόσο εύκολο ήταν να δημιουργήσεις/ να ενωθείς με τον παιχνιδι;

Πολύ Εύκολο 1 2 3 4 5 **Πολύ Δύσκολο**

8) Πόσο κατανοητό ήταν να δηλώσεις το παιχνιδι; (Μονο ο παίχτης που «έγραφε»)
Καθόλου 1 2 3 4 5 **Πάρα Πολλή**

9) Πόσο εύκολο ήταν να «γραψεις»/ δηλώσεις το παιχνιδι; (Μονο ο παίχτης που «έγραφε»)

Πολύ Εύκολο 1 2 3 4 5 **Πολύ Δύσκολο**

10) Πόσο κατανοητό ήταν να δείξεις την δήλωση σου;

Καθόλου	1	2	3	4	5	**Πάρα Πολλή**

11) Πόσο εύκολο ήταν να δείξεις την δήλωση σου;

Πολύ Εύκολο	1	2	3	4	5	**Πολύ Δύσκολο**

12) Πόσο κατανοητό ήταν να «γράψεις» τις δηλώσεις; (Μονο ο παίχτης που «έγραφε»)

Καθόλου	1	2	3	4	5	**Πάρα Πολλή**

13) Πόσο εύκολο ήταν να «γράψεις» τις δηλώσεις; (Μονο ο παίχτης που «έγραφε»)

Πολύ Εύκολο	1	2	3	4	5	**Πολύ Δύσκολο**

14) Σε γενικό βαθμό πόσο εύχρηστο και ευνόητο ήταν το περιβάλλον χρήστη με τα συγκεκριμένα γραφικά;

Καθόλου	1	2	3	4	5	**Πάρα Πολλή**

GAMEPLAY

15) Πόσο εύκολο ήταν να καταλάβεις τα controls του παιχνιδιού χωρίς κάποια επεξήγηση;

Καθόλου	1	2	3	4	5	**Πάρα Πολλή**

16) Πόσο εύκολο ήταν να σκανάρεις και να έχεις σταθερή την camera στο σύμβολο; (marker)

Πολύ Εύκολο	1	2	3	4	5	**Πολύ Δύσκολο**

17) Πόσο εύκολο ήταν να ρίξεις ένα χαρτί;

Πολύ Εύκολο	1	2	3	4	5	**Πολύ Δύσκολο**

18) Πόσο ρεαλιστικά ήταν τα αντικείμενα μέσα στο παιχνίδι;

Καθόλου	1	2	3	4	5	**Πάρα Πολύ**

19) Πόσο βολική ήταν η θέση των χαρτιών σε σχέση με το πώς καθόσασταν;

Καθόλου	1	2	3	4	5	**Πάρα Πολύ**

20) Πόσο καλά ανταποκρινόταν η camera και γενικά η συσκευή σας κατά τη διάρκεια του παιχνιδιού;

Πολύ Αργά	1	2	3	4	5	**Πολύ Γρήγορα**

21) Ήσασταν σε επίγνωση των κινήσεων των υπολοίπων παιχτών;

Καθόλου	1	2	3	4	5	**Πάρα Πολύ**

22) Σε γενικό βαθμό, πόσο διασκεδαστικό ήταν το παιχνίδι;

Καθόλου	1	2	3	4	5	**Πάρα Πολύ**

23) Σε γενικό βαθμό, πόσο εύκολο ήταν το παιχνίδι;

Καθόλου	1	2	3	4	5	**Πάρα Πολύ**

24) Σε γενικό βαθμό, πόσο ενδιαφέρον ήταν το παιχνίδι;

Καθόλου	1	2	3	4	5	**Πάρα Πολύ**

25) Σε γενικό βαθμό, πόσο κοντά ήταν το παιχνίδι σε σχέση με το πραγματικό;

Καθόλου	1	2	3	4	5	**Πάρα Πολύ**

26) Κατά την άποψη σας, σε πόσο βαθμό θα μπορούσε αυτό το παιχνιδι να αντικαταστήσει το πραγματικό;

Διαφωνώ Απόλυτα	1	2	3	4	5	**Συμφωνώ Απόλυτα**

27) Αισθανθήκατε καθόλου disoriented μετά που τελείωσε το παιχνίδι;

Καθόλου	1	2	3	4	5	**Πάρα Πολύ**

ΓΕΝΙΚΑ ΣΧΟΛΙΑ

28) Κατά την διάρκεια του παιχνιδιού, ρίξατε κατά λάθος χαρτί που δεν θέλατε;

ΝΑΙ **ΟΧΙ**

29) Σε τι προσανατολισμό (orientation) χρησιμοποιούσατε περισσότερο το κινητό σας;

PORTRAIT **LANDSCAPE**

30) Κατά την άποψη σας, ποιο μέρος του όλου παιχνιδιού ήταν το πιο δύσκολο για εσάς;

31) Κατά την άποψη σας, ποιο μέρος του όλου παιχνιδιού σας άρεσε περισσότερο;

32) Τι άλλα αντικείμενα, αν υπάρχουν, θα συμπεριλαμβάνατε στο παιχνίδι;

33) Τι βελτιώσεις και επιπλέον χαρακτηριστικά θα θέλατε να δείτε σε μελλοντική έκδοση του παιχνιδιού;

34) Γενικά σχόλια

A4. “Closed” and “Double-Closed” Definition

These two terms exist on the bidding round and players can declare them when the three passes occur. “Closed” term is declared by the players when they believe that the final bid declared by the opponents cannot be reached. For example when team 1 declares the bid of 12 of Spades, team 2 can “close” the game if they believe that team 1 cannot win at least 120 points to win the game. If team 1 manages to get 120 points then their winning points are doubled-up and team 2 gets no points, else team 2 gets the points. “Double-Closed” can only occur when a team “closed” a game. This means that the team whose game was closed believes that they can reach the points of the final bid. The final points are then quadrupled.