Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Vision-Based Gesture Interaction in Shared Augmented Reality Environments

Caolan Barry
[redacted]

April 30, 2020

A Dissertation submitted in partial fulfilment
of the requirements for the degree of
MSC (Integrated Computer Science)

Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at `http://www.tcd.ie/calendar`.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at `http://tcd-ie.libguides.com/plagiarism/ready-steady-write`.

Signed: ____________________ Date: ____________________

Abstract

This dissertation explores the capabilities of 3-dimensional interactions in a video-based augmented reality application using currently available, state-of-the-art technologies. A collaborative AR board game is designed and implemented as well as multi-modal interaction methods. The game and interactions are implemented using familiar state-of-the-art technologies for smartphone AR applications including Unity, ARFoundation and ManoMotion.

Firstly, background research in the areas of AR, Collaborative AR, Interactions used in AR as well as interaction usability are presented. The game design and implementation is documented followed by the results recorded from this application. Results of this investigation show that the applications frame rate is significantly impacted due to hand tracking and gesture recognition with frame rates dropping on average by 79% from 53.5 to 11.1 frames per second. The environment that the application is used in presents issues in the performance of both the gesture recognition as well as the hand tracking, with a white texture-less backdrop producing the least number of errors. Further work on this topic include performance optimisation of the application as well as a user study to determine whether the satisfaction and engagement of this interaction is improved when played in a co-located, face-to-face setting.

Acknowledgements

I would like to thank my supervisor for this dissertation, Dr. John Dingliana, for his support, feedback and guidance throughout the year. I would like to thank my parents and siblings for their patience and support throughout my final year in college as well as throughout my undergraduate education.

I would also like to laud the SCSS as well as the wider Trinity College community for their efforts in reducing disruption to student education during this unprecedented pandemic.

Contents

List of Figures

List of Tables

1 Introduction

While augmented reality (AR) isn't a newly proposed medium or technology, the availability and affordability of augmented reality capable devices have meant that it has previously remained a topic of research rather than a mass-market medium. As computers became smaller and more powerful, the computations needed for augmented reality are now capable on smartphones as well as purpose-built augmented reality headsets. Augmented reality gives designers and creators a new medium for games, visualisations and experiences to be consumed. With this new medium, which some consider as the next mass-medium [5], comes a novel opportunity for creators to change how we interact with games and experiences.

Multiplayer video games have existed for over 60 years, with one of the first-ever multiplayer games created in 1958 [42] and have grown to become a multi-billion dollar market in recent years. As video games became more popular on smartphone devices, the interaction method for these devices have largely remain unchanged. With many smartphone devices now possessing the hardware requirements to satisfy a video-based AR experience, new interaction paradigms have become actualised by using the devices camera.

Implementations of vision-based gesture interactions in AR have seen both marker based approaches to detect finger tips as well as marker-less detection of the hand, fingers and gestures. Investigations of marker-based solutions performing tasks such as direct manipulation have proven to be satisfying and engaging for users. I will be investigating the impact the marker-less solution has on the experience for performing tasks such as object selection, direct manipulation and de-selection for a video-based mobile smartphone experience. I will be interested in how this implementation affects the performance of the application as well as the errors incurred from using this application in different environments.

The paper is laid out as follows: firstly background research of the topics are presented for the topics of: AR, games, interaction and usability. Secondly, the design of the game and the application is detailed. The implementation of this design is then documented followed by the results and evaluation of the experiments. Finally, I present the conclusion of this research as well as further work that could be conducted to improve parts of the system.

2 Background Research

2.1 Augmented Reality

Augmented reality refers to the visual medium that overlays virtual objects into the real world. The pioneer of this topic of research was Sutherland, who both conceptualised[57] and built the first augmented reality headset[58]. Since then, Augmented reality has been studied in several domains such as educational tasks[32] [55] [27], tourism[24], medicine[9], and entertainment [50].

The appeal of AR as a medium to consume digital content is that the overlaid visual objects are not only augmenting the user's vision of the world, the objects themselves are registered in 3 dimensions. This is how we naturally perceive information about the world using our sense of sight.

To define AR more formally in the context of this dissertation, I will be using the definition proposed by Ronald Azuma. He formally requires AR to have all three of: *"combines real and virtual, interactive in real-time and registered in 3-D"..* Also, a distinction is made with respect to Virtual Environments, requiring AR to *"..allows the user to see the real world, with virtual objects superimposed upon or composited with the real world"*[7]. Thus, AR must allow for the real world to be visible.

AR allows simulation of real-world objects in 3-dimensional space without actually needing them. This allows someone developing a game, for example, to make the game board virtual while only requiring a table to play upon. Shared experiences of augmented reality immerse more than one user into this hybrid real-virtual environment.

2.2 Shared Experiences in Augmented Reality

Involving more than one user into an augmented reality experience allows multiple users to share an augmented space. The same overlaid digital content can be seen registered in the same space. Collaboration in these environments can benefit from the user's capabilities to see each other as well as sharing this space. The Studierstube[59] was one of the first collaborative AR applications. The goal of the application was to maintain natural human interaction between participants while simultaneously immersing the participants in an augmented space. Here participants can see and interact with 3-dimensional visualisations. Their normal habits of communication remain unhindered, body gestures are seen through the see-through display and they can also speak and hear one another. The Studierstube allowed collaboration in real space with rich 3-dimensional visualisations, 3-dimensional interactions were superior to its 2-dimensional screen and mouse counterpart as full freedom to navigate around the model was possible with communications between participants unhindered. Interaction with the visualisations was enabled through a pen-like 3-dimensional interaction device.

Other collaborative tasks such as design also can utilise the hybrid nature of AR, Kiyokawa et al.[40] conducted experiments comparing shared virtual environments (SVE) and shared augmented environments (SAE). An experiment replicated in both AR and VR is conducted where one participant needs to direct the attention of the other by pointing at a virtual cube. The task is completed when the other participant has clicked on the correct cube. The shared augmented environment saw task completion times were lower than the virtual environment where the participants relied on a virtual dummy that moved in accordance with the other user's movements. Also found was that satisfaction within the augmented environment was higher. Participants noted it to be more relieving to be able to see one another rather than a virtual representation of each other. This study has shown that tasks that require collaboration are more efficient when those we are collaborating with are visible and when non-verbal communication, such as pointing and gestures, are also visible. With accurate enough registration, AR allows collaboration in digital workspaces such as modelling to have sure communication and interaction with zero incurred latency, seeing another participant point, shrug or shake one's head can be seen immediately through the transparent lenses.

Augmented teleconferencing is another topic investigated and improved by Billinghurst et al.[12] that also presents the benefits of greater perception of natural communication. Using markers to register in place, live video and sound of a colleague saw the improved understanding of non-verbal cues. A tangible user interface allowed users to move markers around freely, allowing full control over where one can see the other users in the call. This study has shown that marker-based AR enables real objects to have rich interactions in AR settings with both the interaction and the consequence being visible in the same augmented space.

A Table-top game in AR was developed [13] and was found to be not only be possible in AR but a fun way to play these games. Difficulties in imprecision were overcome with a geometrical snapping of faces and planes of objects such as the board-pieces or cards. The players were able to play the game in a precise way with an imprecise input mechanism. This type of collaborative experience, however, had each independent viewer view their own instance of the game. Therefore the co-locality of the players wasn't necessary, and the virtual content didn't occupy the same world space.

Daniel Wagner et al.[66] investigated whether multi-user mobile augmented reality could be achieved using PDA devices with cameras and a wooden miniature track board. The goal was to improve upon existing mobile augmented reality by providing an unconstrained way to use mobile augmented reality. Previous solutions saw users wearing backpacks with pieces of hardware such as laptops and sensors attached to them [24] [37]. A PDA device was used as a balance between portability as well as computational ability. A distributed game was developed that used place markers to determine the PDA's position and rotation in the world with respect to the game board. The game was trialed with many users and confirmed to them that using this unconstrained augmented reality has the easiest accessibility. Users were able to freely exchange the device between them and had no hindrance other than holding the device. This was a feat far less possible using a wearable device of hardware.

A more recent, prevalent use of augmented reality for shared experiences was the mobile augmented reality game Pokemon GO, which has amassed over 800 million downloads to date [31]. This game became a global phenomenon, even leading to mass gatherings in public parks and other locations that had virtual functions inside the game. Pokemon GO was an indicator that people are interested in playing games and consuming media through a new medium, the medium of AR. The game consists of visiting real life locations in order to have a chance of capturing new *"pokemon"* characters which is an objective of the game. These *"pokemon"* are superimposed into the real world using the GPS data of the mobile device to determine the player's position relative to the virtual character. While not all elements of the game are shared and multiplayer, there was a collective experience of attempting to catch them all and train them. This was enough to capture the attention of hundreds of millions of people.

Multiplayer games have existed for thousands of years with some of the earliest board games dating back to 3000 B.C.[11] However games of any kind such as those not requiring any material thing are played by animals as well as children, and the existence of games are tied with the existence of play itself, which Huzinga describes as *"a function of the living, but is not susceptible of exact definition either logically, biologically, or aesthetically"* [36]. The existence of gaming is therefore inevitable and is a necessary part of our lives. It satisfies a need to play and cannot be logically justified.

Augmented reality opens up a new way to both see and interact with games. First and foremost, true AR has 6 degrees of freedom for the user immersed in the experience. This means players have the freedom to change their geometrical position in world space as well as where they are facing. A real AR experience would allow its user to navigate freely, changing their world position (latitude, longitude, altitude) as they wish and for the digital objects to remain registered or anchored in place. The second three degrees of freedom is allowing the user full control to change the rotation of their viewport. This can be thought of as the angle and direction of the head in the case of a head-mounted display. For a mobile device, the angle and direction of the camera, smartphone or PDA device. These combined give the user full immersion inside the augmented space, where they can view augmented objects from any point in that space.

2.3 Registration

Registering digital objects in 3-dimensions is achieved generally in two distinct manners: marker-based registration and marker-less registration. Both approaches rely on imagery from a camera device which uses computer vision algorithms to detect features in the scene.

2.3.1 Marker-Based

Marker-based registration begins with selecting an image or fudicial that is known to the developer. This is usually in the form of a printed QR code or a shape with distinct geometry. The first fully self-contained backpack-less augmented reality mobile application developed for a PDA device used ARToolkit to determine the pose of the camera relative to markers placed among walls. The ARToolkit library was ported to the PocketPC platform.The system architecture was a hybrid architecture that allowed the device to offload the camera images to a server to handle the vision aspect if necessary[67]. This was a great stride toward handheld mobile AR that requires no extra hardware or sensors. It relies purely on vision algorithms to solve the registration problem, and it does this without carrying any other computational device such as a laptop that was seen by the first fully mobile AR application[37].

2.3.2 Marker-less

Marker-less AR, as its name implies, doesn't require the use of a pre-determined image to solve the registration problem. Instead, it can use other sensors such as GPS and a sensor for orientation[37]. Out of the box marker-less solutions offered by high-level scripting API's such as ARCore, ARKit and ARFoundation use a combination of sensors to allow detection of planar surfaces visible by you. They use feature tracking for each frame produced by the camera. This, combined with accelerometer data, is used to determine the change of the position of the device. As the device moves and captures the scene from different perspectives, a feature matching algorithm can detect the same feature from various aspects. This allows feature registration between frames and subsequently position and depth estimation to be determined relative to the camera. Features that are consistent with a plane-like structure can then be used to solve the 3-dimensional registration issue[6].

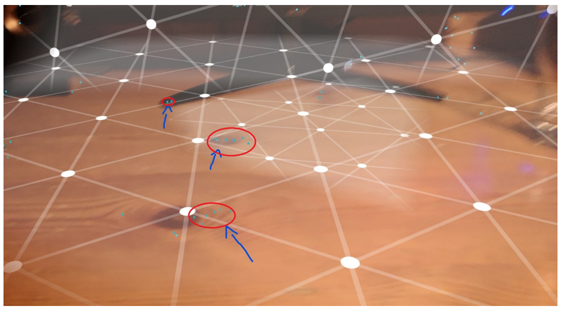


Figure 2.1: Feature Point Detection

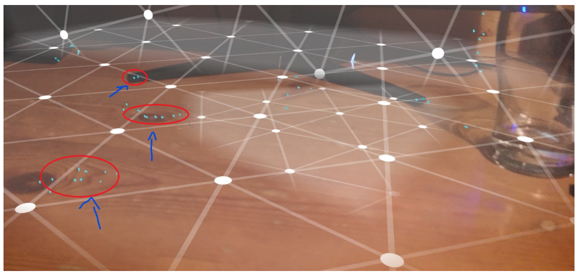


Figure 2.2: Feature Point Matching

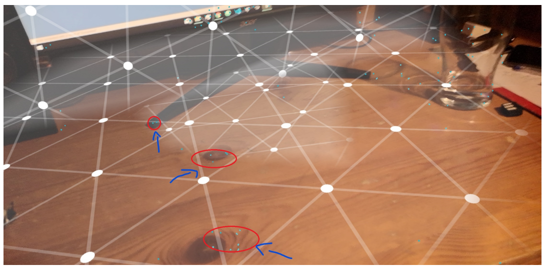


Figure 2.3: Feature Point Matching 2

2.4 Interaction in Virtual Environments

Virtual and Augmented environments have presented themselves as impressive candidates for 3-dimensional visualisation, modelling and gaming, however, the 6 degrees of freedom experienced by the user may suffer if they are stuck using a 2-dimensional input device. Direct manipulation and interaction with objects becomes desktop-like and inherits issues from the occlusion of one virtual object by another. A metaphor used by Zimmerman [69] is that a 2-Dimensional input device pointing at a transparent bowl of cherries is not inherently able to determine whether you intend on interacting with the bowl or the cherries inside the bowl. It mandates the un-occluded line of sight between the user and the target. 3-dimensional input, however, will allow one to *"pick up the desired item, without ambiguity"*.

2.5 Gesture Interaction

Gesture interactions allow the motion of the user's body to interact with a computer system. This interaction type falls into the category of a natural user interface, an interface that seeks to emulate how humans interact with the real world and each other rather than how we interact with technology. When we want to direct attention to an object in the real world, we use our hand and fingers to point in the direction of it. When we want to grab an object, we place our hand on it and use our fingers to grip it. Using our hands to interact in a virtual environment has the same appeal and naturalness as using them outside the environment.

Natural user interfaces have proven to be accessible and usable by the population. In the correct context, they can replace the graphical user interface (GUI) of specific applications that don't require them. Voice assistants, for example, have become increasingly popular in homes as well as on smartphone devices. Accurate translation of voice data to words and sentences can save user's time retrieving a device and typing on them, they are hugely more efficient.

2.5.1 Personal Interaction Panel

The Personal Interaction Panel [60] is a 2-handed, 3-dimensional input device that represents a familiar pen and paper interface. The pad is used for GUI elements in which the pen can interact with. Also, the pen can be used in free space, to interact with digital objects. Although two hands are required for this interaction type, no fatigue was reported by the users. The pad element of this interface could also be removed, leading to just a single 6DOF, single-handed pen interface for interacting. The overhead incurred from magnetically/optically tracking the input device, similar to [69] mean that the use of this interaction is confined to a single room. Configuration of optical/magnetic trackers restrict where this device can be used. Only the pen part of this device would be suitable for a mobile video-based experience as one hand will be occupied by the mobile device. Further, the setup times of this solution may outweigh any benefits it brings.

2.5.2 Glove-Based Interaction

Glove-based interactions were the first to track the position of the hands and fingers accurately. A significant application for gesture recognition has been in sign language recognition[3] [23]. The DataGlove created by Zimmerman [69] had the goal of creating natural interaction for 3-dimensional environments such as an AR environment by adding depth to interaction, something which a mouse cannot achieve. Flex sensors on the fingers of the glove that measure the angle between the individual parts of the finger were used to achieve this goal. Flexing/bending of the finger is measured through these flex sensors allowing determination of which part(s) of the finger(s) has moved. Determination of the location of the hand itself relies on two ultrasonic transducers as well as three receivers which approximate its location using triangulation approximations. This device was used as a gesture-based controller for presentations by Thomas Baudel et al. [10]

Among the findings were that having a glove-based implementation for gesture recognition mandates specific characteristics about the user's hand, the most detrimental being the size of the hand. A mismatch of glove size and a users hand resulted in 2 test subjects only achieving a 50% recognition rate of their gestures. After a short demonstration of the gestures, the rest of the subjects landed between 72-84%, and users who had become familiar with it achieved between 90-98%. The glove relied on wired sensors that are ultimately connected to the device running the application; this brings limitations in the freedom of movement the user has. It was also deemed an uncomfortable device.

Glove-based gesture recognition has proven to be effective for users who have suitable sized gloves as well as some practice with the interaction. This excludes the device from being a viable interaction for those without a fitting glove. Tethering the glove to the machine running the application also has an impact on satisfaction. The illusion of natural interaction using common gestures such as swiping pages is now marred by wielding a glove on a wire that feels unnatural.

A comparison of glove-based and vision-based gesture recognition for both static and continuous gestures by Oinam et al. [15] found glove-based interactions less accurate. This study used a flex sensor glove similar to the one aforementioned and was compared to a vision-based technique for reading gestures defined by the number of fingers not being clamped to the palm. In this context, a vision technique using feature extraction and canny edge detection is used to count the number of fingers in the image. This study showed that vision techniques can achieve an extremely high level of accuracy, in this particular study 100%. However, the vision based techniques were performed against a white backdrop with a directional light to provide a high level of clarity for the images, which lead to the conclusion that this technique *"can give an accurate result only and only if good lighting condition is provided with a stationary white background"*. Glove-less interactions are, therefore, only suitable in contexts where both high contrast and high brightness are satisfied.

2.5.3 Vision-Based Interaction

Glove-less techniques require the subject of the interaction, in this instance, the hand, to be in sight of a camera device. Different methods for extracting the correct information from the hand have been proposed, some with different requirements than others. The most apparent benefits presented by vision-based gesture interaction is its portability. Requiring no external hardware, it presents itself as the best candidate for 3-dimensional interactions for mobile AR environments. The drawbacks of a vision-based gesture interaction are primarily related to the visibility of the hand. A low lighting environment, for example, would make distinguishing features of the hand difficult and possibly inaccurate. Self-occlusion of the hand from the perspective of the camera can also lead to less useful information extracted.

Freeman et al. [26] proposed a vision based method to be used as a controller to play video games. The hand and body were both tracked. A simple algorithm had to be used as the algorithmic cost of a more sophisticated algorithm would incur too much latency. They achieved a response to a user's movements within 10ms. These methods for interacting with games have also been seen on commercial home console systems such as the EyeToy for the Playstation 2 or the Kinect for Xbox 360. Vision-based approaches at detecting hand gestures can be categorised into static and continuous gestures. A review of vision-based hand gestures [46] distinguishes gesture recognition into two distinct steps. Firstly, feature recognition is performed on the image to extract the relevant features such as the edges of the hand and fingers. Classification seeks to classify the features or model of the hand to a specific gesture.

- Feature Recognition
    1. Kinematic Model Approaches
    2. View-Based Approaches
    3. Low Level Features Based Approaches
- Classification
    1. Rule-based approach
    2. Machine Learning

The kinematic model [56] approach uses a model of a hand which is used to search for the kinematic parameters that matches a 2D projection of this model with the features of the hand extracted from the image. The challenges with this method generally lie with how texture less the skin is requiring high contrast and homogeneous backgrounds in order for this extraction method to achieve high recognition.

Low level features based approaches solve the feature extraction issue using a simpler method than trying to recreate a model of the hand. Extracting a binary image of the hand against the background is used by [15] where a white uniform backdrop is used simplifying the feature extraction, while [47] used a process of setting a saturation threshold to the image to extract the hand from the background. The latter study extracted images on a non-homogeneous backdrop with background items present. The resulting binary image can be used to extract the position of the centroid of the hand as well as determining the number of fingers protruded to infer gestures.

Rule-based approaches for classifying the gesture from extracted images usually relies on conditions being met for a set number of frames. This is seen in [18] where continuous gestures such as waving and clapping must satisfy rules for typically N = 10 frames. Rule-based systems relies on the ability of humans to define an adequate set of rules for recognising gestures.

Machine learning methods for classifying gestures have been effective for continuous gestures [68]. Hidden Markov Models using both temporal and spatial data to classifying the gesture benefits from where the rule-based approach falls short, human encoded rules aren't required.

2.6 Gesture Interaction in Mobile Applications

The use of hand gestures for interacting with smartphone AR applications has been investigated for many reasons. A form of body language, hand gestures are a natural communication mechanism used in everyday human life. Secondly, AR smartphone applications typically implement multi-touch displays for user interaction, and as a result, this can be prone to errors if the user accidentally moves the device such that the touch interaction misses the target object within the virtual environment. As the virtual objects are registered in the real world, and as the images from the camera move according to the user, so will the virtual objects.

Therefore, any premature movements by the user during this process will not result in a successful interaction, as the image will not be in a relatively stable position on the multi-touch interface. By contrast, the implementation of hand tracking in conjunction with registration of gestures abstracts the need for both hands to maintain stability and accuracy to one hand. With perfect hand tracking, the hand pre-occupied with holding and stabilising the device doesn't interfere with the interaction unless the other hand is no longer visible. Regardless of unintended movements of the smartphone device by the user, a hand maintains the same position within the virtual environment using vision-based gesture interaction.

Gesture interactions in smartphone AR has been achieved previously through marker-based solutions by Hurst et al [38] as well as Loubna et al [4], in order to demonstrate marker-based gesture interaction for various task such as selection and manipulation of virtual objects.

Hurst et al's user studies concluded that the gesture method proved more difficult than touch screen counterparts but participants found it more engaging and fun. They noted that this interaction method wouldn't be suitable for accurate or time critical tasks but could be used as a suitable interaction method for gaming where effectiveness and efficiency mightn't have the largest weight in determining usability of the interaction, noting that some games are purposefully tricky to interact with making mastering the game challenging and engaging.

User studies by Loubna et al. [4] also concluded that it was a fun and engaging interaction and it was also not physically fatiguing, although this experiment was translating objects resting on a table. Billinghurst et al. [8] proposed interaction methods in AR that overcome a user's inability to maintain a perfectly steady hand while interacting. An implementation of freezing the frame temporarily as well as a fingertip tracking gesture interaction were implemented, the fingertip detection method scored highest in fun and engagement but was considered the most physically stressful and had the least performance.

Methods for capturing the whole hand to be used as an interaction method have been proposed by Park et al. [16][54] using pose estimation of the palm allowing gesture interactions from opening and closing ones hand.

AR has also been used to augment tabletop games using tangible interfaces to compliment real world game pieces with virtual graphics. Molla et al. [45] used a webcam to compliment a monopoly board by overlaying 3D graphics to compliment the view of the game. Ta et al. [61] took this a step further and used a tangible interface of tokens and grid cells to play a collaborative AR game.

2.7 Interaction Evaluation

Interaction design is the process of designing interfaces between computer systems and the end-user. Interfaces of a computer system are the key to allowing humans to instruct systems to perform a task. For personal computers, the windows, icons, menu, pointer (WIMP) model of computer interaction has become the standard model since the Xerox Star in 1981 [14]. For mobile devices and PDA devices, a similar interaction is also the case, with the pointer of a mouse replaced by touch interaction. With multi-touch displays, more interaction paradigms were actualized, such as pinching gestures to change the scale of images and documents[64]. The biggest and most central concern for designing interfaces as well as the interactions used with these interfaces is their usability[53].

Usability seeks to measure how usable the device is to the user, for without the ability for a computer device to be used, its performance, capabilities and functions quickly become useless. The International Organization of Standards defines usability as *"the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use"* [1].

Researchers in the field have more specific characteristics of usability. Nielsen defines usability as requiring five quality components: Learnability, Efficiency, Memorability, Errors (rate) and Satisfaction[29]. Quesenbery proposes using the 5Es of usability: Effective, Efficient, Engaging, Error Tolerant and Easy to learn,[52] while Preece, Rogers and Sharp define usability in terms of goals: Efficient to use, Effective to use, Safe to use, Good utility, Easy to learn and Easy to remember[53].

Effectiveness is described as *"whether the software is useful and helps users achieve their goals accurately"* [52], *"how good a product is at doing what it's supposed to"* [53], *"accuracy and completeness with which users achieve specified goals"* [1]. Nielsen doesn't include effectiveness; however, he defines utility as *"whether it provides the features you need."* [29].

Efficiency is described as *"Once users have learned the design, how quickly can they perform tasks?"*[29], *"the way a product supports users in carrying out their task"*[53], *"the speed with which work can be done"*[52], *"resources used in relation to the results achieved"* [1].

Errors are raised explicitly by [29] and [52], while [53] uses the metaphor of safety to describe a system designed to incur the least amount of errors. A safe system is described as one that has been designed such that there is the least amount of risk posed to physical safety due to human error, for example, in aviation or energy production. But this also applies to the ability of a system to tolerate and recover from human error where physical health isn't necessarily at risk but files on a file-system may be at risk of corruption/deletion which incurs an expense of resources such as time, money and labour. A safe system is one that is designed such that users make the least amount of errors while interacting with it and those errors made don't incur a high cost of resources to return to the same point.

Learnability and memorability are included by [53] and [29], while Quesenbery [52] only mentions learnability. Learnability is the measure of the time it takes for a novice user to become proficient at using the system from the beginning of their first interaction with the system. In contrast, memorability measures the time to proficiency of the same system after a period of time, without having interacted with the system. A memorability time the same as the learnability time is not memorable at all.

Satisfaction is mentioned by [1] and [29], while [52] uses engaging to describe *"how pleasant, satisfying or interesting an interface is to use."* [53] doesn't explicitly mention satisfaction and perhaps for a good reason. It is unlikely that a system with poor efficiency, effectiveness and learnability is going to be highly engaging and satisfying for the user, although not impossible. Satisfaction is also a subjective quality which [53] does note. A system that is highly inefficient requiring extraneous effort to interact with such as requiring typing "yes" or "no" to questions rather than selecting one of the two options, which provides minimal effectiveness at completing your task and also has no tolerance to errors, requiring a full walkthrough from the beginning when an error is made, which is likely to frustrate the user and waste their time leading to poor satisfaction.

Usability is therefore multi-faceted, with some interactions more usable than others for different reasons. Some may be more efficient at performing a task, while others may produce a higher level of satisfaction or effectiveness. The context of the application must also be considered when evaluating usability. Gaming may not pose a specific task to be completed by the intended user other than to play and enjoy oneself. Engagement and satisfaction would be of the utmost importance for designing interactions for games. Some games have interactions that are tough to learn and difficult to master, yet are still enjoyable to play contrasting to traditional systems not in the domain of gaming.

Evaluating the interfaces that we have designed and implemented is key to finding issues with the usability of the system. After gathering requirements, designing and implementing an interface, an evaluation will present any issues not accounted for in the previous stages of the interaction design process. Usability evaluation methods have been categorised into Inspection methods and test methods. Inspection methods differ from test methods as they don't require involvement of end users in the process.

2.7.1 Heuristic Evaluation

Heuristic Evaluation is an inspection method that has become widely used in usability testing. First proposed and later refined by Nielsen[48], it follows multiple evaluators inspecting the interface in isolation of each other and documenting any usability issues they encounter during the lifecycle of their interaction with the system. Nielsen proposes 10 usability heuristics [48] but category-specific heuristics can also be applied for applications that provide a similar goal or service. This inspection method seeks to consolidate found usability issues at any stage of implementation of the interface. Evaluators should document their opinion about what part of the interface is violating a heuristic. The compiled list of apparent violations are used as a guideline for what to avoid for the next iteration - they do not always present a definitive solution, rather advise what should be avoided on the next iteration.

Nielsen also examined the cost-benefit of the number of evaluators involved in the process, where experiments found that a single evaluator would find on average 35% of usability issues with the optimal number of evaluators ranging from 3 to 5 and any additional evaluators yielding diminishing results. Extensions of these heuristics have been proposed in the gaming software domain.

2.7.2 Cognitive Walkthrough

The cognitive walkthrough method is an informal inspection method similar to a heuristic evaluation that follows an evaluators step-by-step set of actions while using the system to complete a given task [34]. It is used to simulate a first-time user's experience of using the system and it's goals are to improve the learnability of the system. The advantage of performing a cognitive walkthrough is that designers can take on the role of an end-user and identify possible issues. The disadvantages of a cognitive walkthrough is that improper tasks may be selected and lead to a biased walkthrough, thus not receiving valid feedback on what issues are present with the learnability of the system for a novice user. Its also difficult to predict how exactly a user is going to interact with the system.

2.7.3 User Studies

Involving users in the evaluation of the usability of the interface is a formal method for gaining feedback on the interface as well as the application it is designed for[29]. Using representative users for the application, users should be asked to perform a task with the system and an observer should observe their interactions. A similar method should be used for video games, although some note that performing a task analysis on games isn't very suitable as there isn't an inherent task other than to play[51]. This does apply to certain exploratory games, but for multiplayer competitive simulation games, there is often a repetitive task to complete to defeat the other player. As a user study was unable to be completed for this dissertation, the further work section proposes an appropriate user study.

2.7.4 Usability in Video Games

Heuristic Evaluation for Playability (HEP) for video games are an extension of Nielsen's HE and have been proposed [19] and extended [20] by Desurvire et al. who conducted both a Playability heuristic evaluation as well as traditional user testing. By using an extensive list of heuristics in the categories of *game play, game story, mechanics* and *usability*, they found that evaluation using these heuristics allowed them to detect more usability issues than their user experiments. There was also a large amount of overlap between the issues found from both these methods, suggesting that a lot of these issues could be solved before involving users in user tests. They conclude that user tests are not replaceable as human behaviour can be unpredictable.

Pinelle at al. [51] compiled 10 heuristics focusing solely on evaluating the usability of games where Desurvire [19] provides heuristics that encompasses other parts of game design such as the narrative: *"Player understands the story line as a single consistent vision"*. They used Dykstra's [22] method of creating domain specific heuristics by evaluating existing products. 108 reviews were analysed from a popular game critic website and 12 categories of usability issues were formed. 10 heuristics were formalised from these categories and were used in a game evaluation. Comments from the evaluators included *"nearly everything that is frustrating about the usability of the game tested was easily identified with a heuristic"*. These heuristics were confined to PC and similar console games, they note that for input devices such as the one used with the Wii where motion is used, new heuristics may need to be considered.

2.7.5 Usability in Augmented Reality

Dunser et al. [21] apply HCI principals to the medium of AR. Mentioned are usability principles seen previously: learnability, error tolerance and learnability, as well as principles that have specific relevance for AR systems and interfaces. Low physical effort indicates that a user can complete an action with the minimum amount of fatigue occurring from the interaction. Flexibility in use should be adhered to in order to allow the user different methods of interaction, this accommodates for the user's specific preferences. Multi-modal interactions should therefore be implemented. Error tolerance is of high concern for AR applications as they are quite prone to instability. Usability heuristics have also been adapted for the medium of augmented reality by [41].

The virtual environment of AR applications have afforded their users new methods of interacting with the environment, vision-based gesture interactions provide hardware-less natural interactions that have produced higher satisfaction and engagement with users while sacrificing completion times for tasks. As AR is currently presenting itself as a new medium for creators to create games and simulations in virtual environments, natural interaction methods involving the users hands have been actualised, this dissertation investigates the capabilities of 3-dimensional interactions in a video-based augmented reality environment. A collaborative AR board game is designed and implemented with vision-based 3-dimensional interactions. The game design requires either a marker or flat-surface to play on with no additional hardware for interaction or objects used as a tangible user interface.

3 Design

This section covers the design of an AR application suitable for a smartphone device. This design supports the goal investigating how marker-less gesture interactions impact the games performance, this is achieved by designing multi-modal interactions that the user can switch between. The design of the game is also detailed in relation to selecting, manipulating, transforming and de-selecting objects within the virtual environment. To summarise the design of this game, it closely resembles the Connect4 tabletop game as it is played in real life while also permitting a real-time augmentation of the game to replace the turn-based nature.

3.1 Game Overview

A multiplayer, real-time augmented reality game was designed heavily inspired by the classic tabletop game Connect 4. This game is played by two users each having their own set of game pieces, red and yellow, competing with each other to be the first player to have four consecutive pieces in the game grid in either the horizontal, vertical or diagonal positions. This game is traditionally a turn-based game, but a design decision to implement the game directly as it is played normally without any restrictions regarding turn allows the game to adopt new characteristics that rewards speed and precision as well as the usual strategy of the tabletop classic. Therefore, the design has replaced the turn-based aspect of the game with real-time freedom for either player.

This augmentation allows this game to transcend from a multiplayer turn-based game to a multiplayer real-time competitive game. The objective of the game remains the same, except the players now must watch their opponents actions at all times to detect their intended interactions. A player must now consider both the state of the game board as well as their opponents intended action at any given time. This differentiation means that the state of the game can be changed at all times by either player.

The game will conclude when one player has beat the other player to connect four pieces or will end in a draw if neither player has managed to satisfy this winning condition. The game is designed to be consumed on a modern smartphone with a camera, enabling a video-based experience.

3.2 Game Design

3.2.1 The Game Board

The game board used in a Connect 4-like game is a grid-like structure consisting of six rows and seven columns. The grid is in an upright, vertical position and is held rigid by supporting stands on either side attached to a flat surface at the bottom of the board used for resting the game on a tabletop. Each column of the grid is hollow to allow for pieces to move freely in the vertical direction and each column is separated from each other such that a piece cannot move in a horizontal direction, i.e. change its row.

The grid is open at the top for each of these columns, usually having a width and depth consistent with the pieces used by the players to be placed in the board. This confines the piece to a narrow column that it is permitted to freefall within. This also hinders its ability to rotate or tilt.

Each cell in the grid has dimensions consistent with the height and width of the pieces. Two parallel holes smaller than the width of the piece at each side allows the players to view the game state at all times. This details what cells are free and what cells are occupied and by whom.

3.2.2 Physics

The game design must simulate the physics of the real world to allow for the placement of pieces and the collision of pieces with each other. As this game will be played in an augmented space, virtual objects should observe the same physical phenomenon that real objects do at the macroscopic level. In the context of this application, Newton's formative three laws of motion[39] are sufficient to maintain the illusion that the virtual content exists in real space.

These laws are needed to simulate the resting of the game board on a tabletop or flat surface. The pieces need to undergo acceleration caused by the phenomenon of gravity to fall to the last free cell in each column. If there is no other piece in the column it was placed in, it will fall until it reaches the bottom slot of the column where it collides with the bottom of the grid. Subsequent pieces placed in this column collide with the previous piece, causing it to rest in the cell above that piece in that column.

Players should be allowed to grab, drop and place pieces as they choose and will be able to see the opponents' previous successful actions on the board as well as their current action if they are currently manipulating a piece. As the pieces that are within the game board are already placed and are only free to move vertically, these pieces will need to be designed so that a force cannot be exerted on them.

Facilitating the accurate placement of game pieces into slots on the game board can be achieved using a *"snapping"* mechanism. This can aid the player in placing pieces into the top of the board. This has also been employed by [13] as well as a similar alignment aid used in [40]. The design should accommodate for small inaccuracies of interaction so that they don't lead to frustration and dissatisfaction. The original version of this game requires accuracy in the order of millimetres and may be too difficult to achieve in a virtual environment.

The snapping mechanism will also temporarily take control away from the user for a given region of space indicating to them that they are free to drop it here and their action will be successful.

3.2.3 Multiplayer Aspect

Connect 4 is inherently a multiplayer game with two distinct opponents. Therefore, the game is designed to allow for multiple players playing and interacting simultaneously. Removing the turn-based aspect of the game that is developed, therefore, implies that using a single instance of the game on a mobile device wouldn't be sufficient, therefore this design mandates a distributed multiplayer architecture.

Separate instances of the application on separate smartphone devices is required, with the screen of each device used as the medium to consume/view the game as well as the controller for playing it. A networked approach for synchronising the state of the game is required. Different architectures for distributed multiplayer online games include peer-to-peer, client-server and mirrored-server architectures. As each game will only have two players playing simultaneously, a client-server architecture presents the best for consistency[35]. The concerns of poor robustness and scalability aren't of real interest in the context of this application as each managed game will only be serving two clients. Also latency costs from this architecture that arise from a high number of clients are not of concern as two clients will be the maximum permitted players for a given game.

The server's responsibility within the architecture is for transmitting updates about the state of the game board between the two clients as well as the position, rotation and velocity of the game pieces in the environment. Each client (player) can move their own set of pieces; these movements will be sent to the server and subsequently to the opponent's client.

3.2.4 Augmented Reality

The augmented reality aspect of this application is to register the game board and pieces in the real world. The application is designed to support both a marker-based and marker-less based registration. Marker-based registration allows two co-located players to view a board anchored in the same position, giving the players a hybrid view of the interaction that the other player is attempting on their device as well as the consequence of that interaction through their smartphone. Body language and other non-verbal communication are unhindered from this design. Once the game board is in a satisfactory position, the user should be able to lock the board in that position permanently, registering it at this point in space for the duration of the game until the game has ended.

3.3 Interaction Design

To design a usable interaction for a game, one must consider who the target of the application is, where it will be used and how they are going to be used[53]. The interaction will be designed for anyone with the competency of a multi-touch display, such as those commonly found on smartphones, they will also require unrestricted movement of their arms, hand and fingers for the gesture interaction. Where the application will be used is quite versatile. Being designed for a battery-powered device and not requiring additional hardware or controllers, it can be used geographically anywhere. How the user interacts with the game is the topic of my investigation, and therefore, multiple interactions will be designed to move the pieces including familiar interaction paradigms for multi-touch displays as well as a 3D gesture-based interaction. For the specification of the game, the player will be using direct manipulation to move pieces to the top of the board and drop them in their desired slot. The requirements needed are a **selection** of the desired piece, **manipulating** its position and **de-selecting/dropping** the piece. As mentioned in Section 2.3, a mobile handheld augmented reality application will require at least one hand for holding and directing the device. Therefore all interactions are possible with one hand. Interaction feedback is used to reduce the gulf of evaluation after selection and dropping of a piece.

3.3.1 Raycast Interaction

The first interaction is a familiar direct manipulation interaction. Synonymous with the drag and drop interaction of WIMP interfaces present on desktops, the target of the interaction must be in sight of the camera and present on the screen.

When the screen receives a touch interaction on the pixels displaying a piece, the user can freely manipulate the position of this piece using both the screen of the device as well as motion of the device within the environment.

Changing the third co-ordinate of the location of the piece, the depth component, is not achievable directly on the screen of the smartphone using single-touch interaction. A locked or constant distance from the player must be used to determine this component. Navigation of the user through the environment is required to add the depth aspect of the interaction.

3.3.2 Pinching Interaction

To overcome the limitations of the first interaction, a second interaction is proposed that takes advantage of the multi-touch display. It augments the first interaction and allows a second finger to allow depth to be variable rather than constant. Multi-touch interfaces were first presented to the world by Jeff Han [64] and have become an integral interaction type between humans and computers on devices such as smartphones, tablets, laptops and some desktops.

Users can optionally use their second finger to add or remove depth voluntarily, allowing them to iteratively add/remove depth, check whether they are at a satisfactory distance and repeat the interaction as desired.

3.3.3 Gesture Interaction

The gesture interaction is designed to emulate as close as possible how this game would be played with real objects. Direct manipulation of objects by the hand is achieved by applying force(s) between the fingers and the thumb, allowing us to bind or hold an object in place relative to our hand.

Designing a natural interaction for manipulation of an object that can fit in our hand in a virtual environment should intuitively follow our interaction in a real environment. A gesture of the fingers and thumb to grab a piece should instantiate the interaction, the piece should now follow the location of the contact of the finger(s) and thumb and should conclude when the contact between the thumb and finger has stopped. The piece should follow the position of the hand in all 3-dimensions.

4 Game Implementation

4.1 Game Board Model

The 3-dimensional model for the game board as described in Section 3.2 was built using Blender, Blender is a free, open source 3D creation pipeline[25]. The implementation consisted of first modeling a single cell of the game board. This was achieved by utilising the pre-built 3-dimensional primitive objects in Blender. Firstly, a cube object with a reduced scale in the Z-direction would give a near-flat square piece. In order to hollow a circular shape out of the cube, I added a cylinder positioned in the center of the cell and used a Boolean difference of the shapes to achieve my goal. This cell was then duplicated into a 6x7 grid as per it's design. A cuboid shape was placed on both sides of each column as well as below the bottom row.

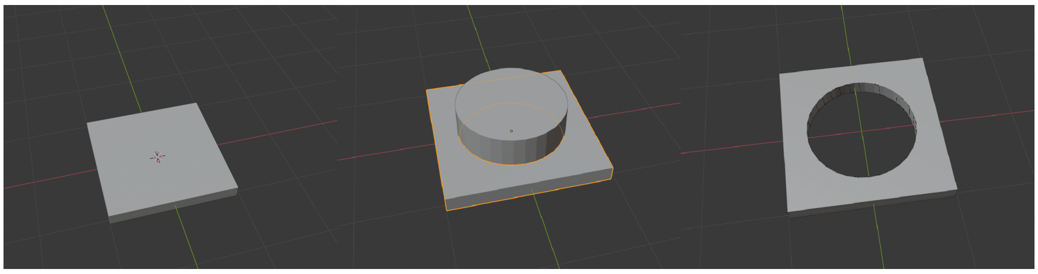


Figure 4.1: Modelling The Cell

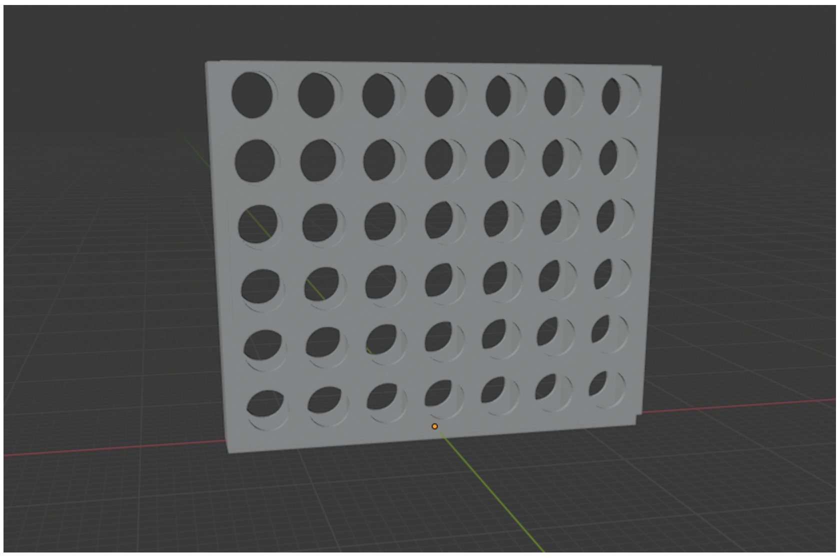


Figure 4.2: Game Board

4.2 Graphics and Physics Engine

Unity3D and the Unity Engine both provide out of the box solutions for creating 3D applications with rich graphics. It has a number of features for virtual environment creation and development such as:

- Importing 3-D models in a variety of different formats/file types.
- A 3-dimensional sandbox for adding, removing, moving, rotating etc. game objects within.
- Support for object hierarchies.
- Support for scene creation.
- Prefabs for various common game elements such as UI.
- High-Level scripting interface.
- Modular components for Animation, Physics, Colliders, Materials etc.

4.2.1 Scenes

I implemented the game in Unity using 3 distinct scenes. Scenes are logical boundaries between separate virtual environments in an application that can be navigated between. In game design, changing scene could be used to transition a player from the end of one level to the start of the next one. When a scene transition occurs, the game objects that exist in memory in the current scene are destroyed/freed and the game objects in the chosen scene are instantiated. This abstracts away extraneous logic and allows development and testing of scenes in isolation of each other.

The first scene in the application prompts the user to input an alias to use for this instance of the game as well as presenting the option to create a game room or join an already created game room. The multiplayer design of the game allows more than one concurrent game to be played at a time. After creating or selecting a room, they then transition to the loading scene as the gameplay scene begins loading and subsequently brought to the gameplay scene when loading has complete.

The gameplay scene begins by prompting the player to decide between marker-based anchoring or marker-less anchoring. The game board will be placed on the marker or on a found plane decided by the user respectively. After two players are present in the game and after placing the board to their desire, the user can then start the game. This scene ends when the game ending condition is satisfied where they can then return to the Room Selection scene to play again.

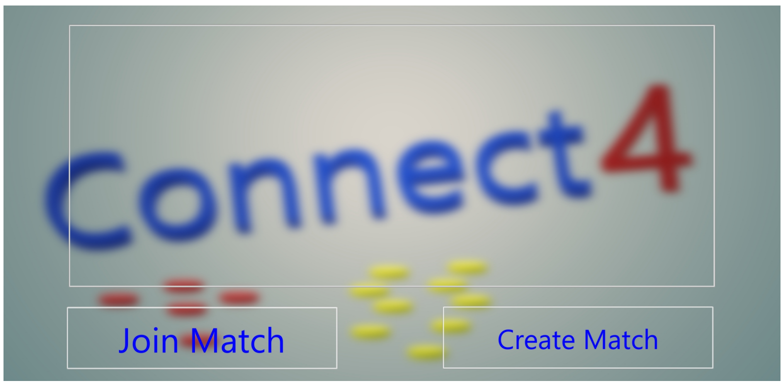


Figure 4.3: Game Room Selection

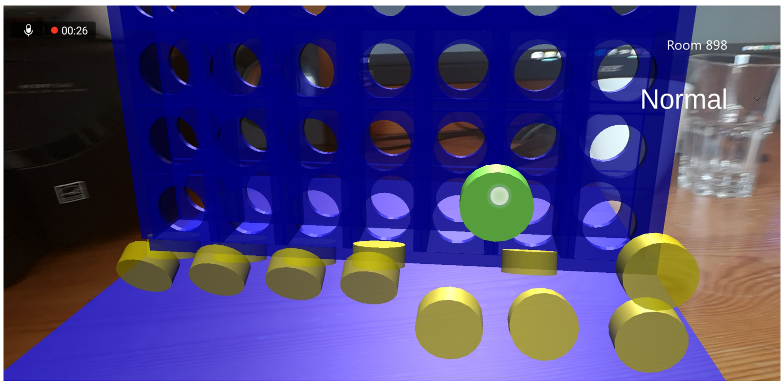


Figure 4.4: Gameplay Scene

4.3 Scripting API

Unity Engine uses an implementation of Mono runtime for scripting: Mono runtime is a cross-platform virtual machine that can be either embedded into applications or used as a stand-alone application[2]. C# is used to extend applications running on the Mono runtime resulting in C# being used as the default scripting language used in the Unity Engine. Unity provides a MonoBehvaiour Class that can be inherited by the user defined scripts to allow for implementing common game logic such as adding logic to the "update loop" of the game, this function is called every frame of the game. The fixedUpdate method can also be inherited from this class, this works similarly to the update method, but should be specifically used for physics updates such as collisions, moving game objects etc. Fixed updates are independent of frame rates which is it's primary difference to the update loop. This has the benefit of processing the movements of game objects independent of the frame rate, without this a drop in frame rate might make gravity seem weaker for example which is undesirable.

4.4 Augmented Reality

Augmented reality was implemented using the ARFoundation package published by Unity Technologies. ARFoundation provides interfaces to different subsystems for working in AR. Subsystems are a *"platform-agnostic interface"*[62] that allows for cross-platform development for both ARCore (Android) and ARKit (iOS) enabled devices. Both the image and plane detection subsystems were used for the marker-based and marker-less options respectively. A reference image library was created using the Unity Editor GUI, and a QR code image was used as the marker. The tracked image manager script is subsequently enabled after the user has selected this option. This script continuously checks the camera's image for the images in the image library and once found estimates it's position in the environment, with the position of the game board then updated to be at this position.

The plane detection subsystem is invoked when the user chooses the marker-less option. This implementation similarly continuously polls the camera image, detects feature points and subsequently estimates planes in the scene. These planes are displayed to the user and a cursor in the center of the screen is used to place the board on a detected plane. The user then places the board into the environment through a place UI button which locks its position in place and now disables plane detection.

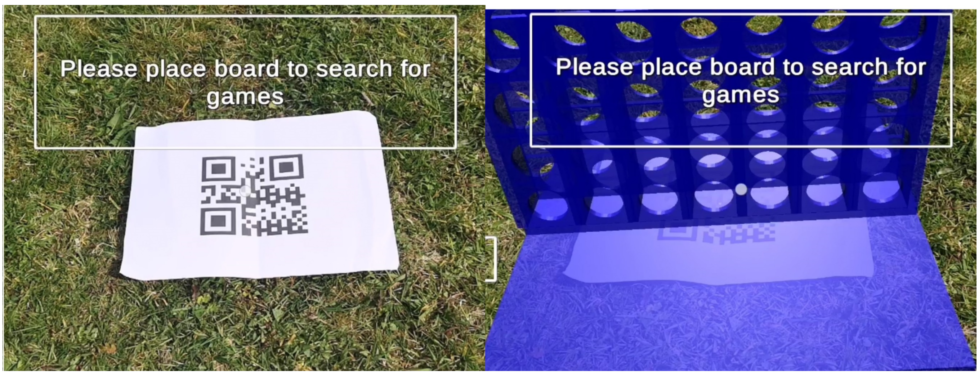

Please place board to search for games
Please place board to search for games

Figure 4.5: Marker Anchoring

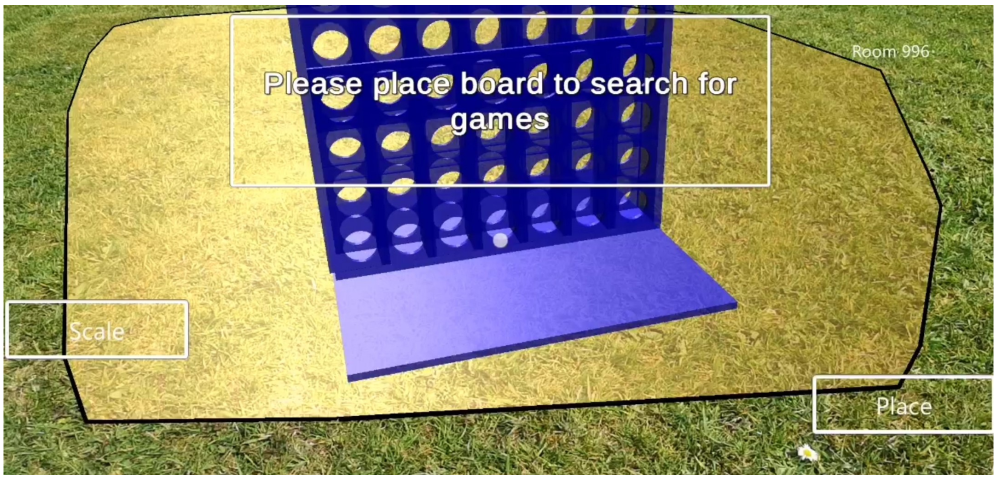


Figure 4.6: Markerless Anchoring

4.5 Multiplayer Aspect

The software library that I am using for implementing the synchronisation of the game state between players is Photon [28]. Photon provides a cloud hosted server for hosting your multiplayer game as well as an API with many out-of-the-box solutions for common game logic such as synchronising the position of game pieces, as well as event-driven message passing. My requirements of this software are to be able to synchronise all positions and rotation values of each players' pieces as well as to synchronise the state of the board. The *"PhotonView"* Class provides methods for writing and reading data from a stream that is transmitted at the serialization rate (50Hz), a piece's relative position data as well as its' velocity and rotation data satisfy the requirements.

An event driven implementation was used for synchronising the state of the board and is triggered when a new piece has been placed in the board, the co-ordinates of the newly occupied cell in the grid is transmitted to the opposing player, both clients then check whether the game ending condition has been met yet.

4.6 Game Synchronisation

Each client instance of the application will have a different world-space. Contrasting to a multi-user application in a completely virtual environment (VE), the player's position relative to the environment cannot be easily determined in AR. This is due to the non-occlusive nature of AR, where it augments our vision of the real world rather than immerses us in a virtual one. In a fully virtual environment, geographic location of the user is discarded and the environment determines where they are. Two users separated by a large geometric distance could be placed facing opposite of each other in a virtual environment and the illusion would work as each others vision of the world is occluded.

The positional co-ordinates of the pieces in this implementation are equal as they are sharing the same world space and synchronisation can be achieved by sending the new co-ordinates. From Figure 4.5, if one player moves the position of the object in the scene by 1 unit in the Z-direction, the new co-ordinates for this game object would be (0, 1, 0). This places the game object at a coherent position in the environment for both clients.

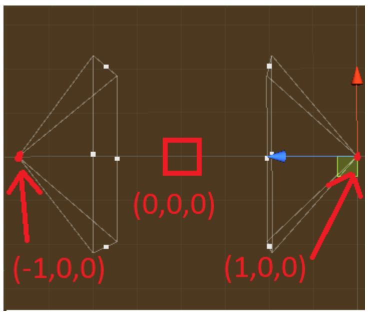


Figure 4.7: Virtual Environment Topology

However, in a video-based AR experience, this same principle should not be used. As the geometry of the real environment is not known by the application, simulating the positions of multiple users relative to the environment has obstacles to be overcome in AR compared to VEs. GPS data has been used previously [37] to determine the position of a user in a hybrid virtual-real environment by recreating elements of the real world into a virtual environment and determining the users position in the virtual environment by receiving GPS updates. This solution for synchronising the position of multiple users would not satisfy the requirements for my application and many other applications as GPS on smartphone enabled devices is usually only accurate within 5m of one's true location at best (under open sky) and can be as inaccurate as up to over 100m for urban environments[65].

A different approach is required in AR to give the illusion to the users that they exist within a common world space. Each client will exist within its own world space with the user (smartphone) at the origin. They then proceed to search for the marker or a plane. Once the user has placed the game board within their own world space, their position vectors will be independent, let $Pos(X) = (x, y, z)$ be the positional co-ordinates of game object x for a given client. Let $Disp(Y, Z)$ be a function equal to the displacement of two vectors such that:

$$Disp(Y, Z) = Pos(Y) - Pos(Z) = ((x_Y - x_Z), (y_Y - y_Z), (z_Y - z_Z))$$

One such way to synchronise common game objects in an AR environment would be to send the displacement of a game object relative to another object rather than its positional co-ordinates. This allows you to programmatically create a virtual origin to synchronise world spaces. The displacement of an object can then be used to synchronise the correct relative place of an object in a scene. In my instance, the positions of all pieces relative to the board will give the illusion that two players using the marker-based implementation were sharing the exact same board and for the marker-less implementation they will see the correct positions of all pieces relative to the board. For two clients (1 and 2), the pieces can be synchronised as follows:

Client 1:

$$Pos(Board) = (2, 0, 3)$$

$$Pos(Piecex) = (1, 0, 3)$$

$$Disp(Piecex, Board) = (1 - 2, 0 - 0, 3 - 3) = (-1, 0, 1)$$

Client 2:

$$Pos(Board) = (1, 0, 4)$$

$$Pos(Piecex) = Pos(Board) + Disp(A, Board)\textit{(from Client1)}$$

$$Pos(Piecex) = (1 + (-1), 0 + 0, 4 + 0) = (0, 0, 4)$$

Each player has ownership over their pieces and is only permitted to move their own pieces. The positional and rotational data is serialized at a rate of 50 times a second, which also coincides with the frequency of the "FixedUpdate" method provided by UnityEngine API. This data is de-serialized on the opponent's client, stored in a Vector3 (Co-ordinates) and Quaternion (Rotation)[63] data type respectively and applied to the game object during the FixedUpdate method call. Each client only subscribes to updates from the opposing client.

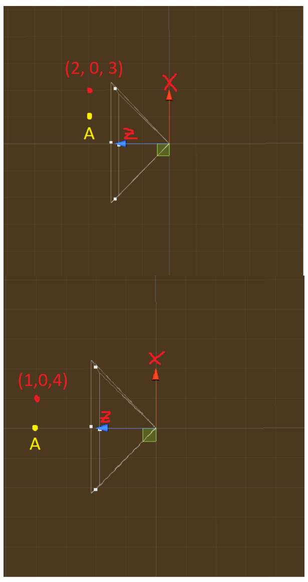


Figure 4.8: Augmented Reality Environment Topology

4.7 Distributed Architecture

The application is compiled to an Android Application Package (APK) which is then subsequently installed onto the operating system of the Android device. Each player requires a smartphone device with this package installed to participate in the game. Photon Cloud is the server hosting service used for this application. It uses the TCP protocol for sending and receiving packets at a user-defined rate. I used the default serialization rate of 50Hz which also coincides with the frequency of the fixed update loop for updating the game scene. Photon API allows for hosting as well as joining game rooms. Game rooms are the individual instances of multiplayer sessions between two players.

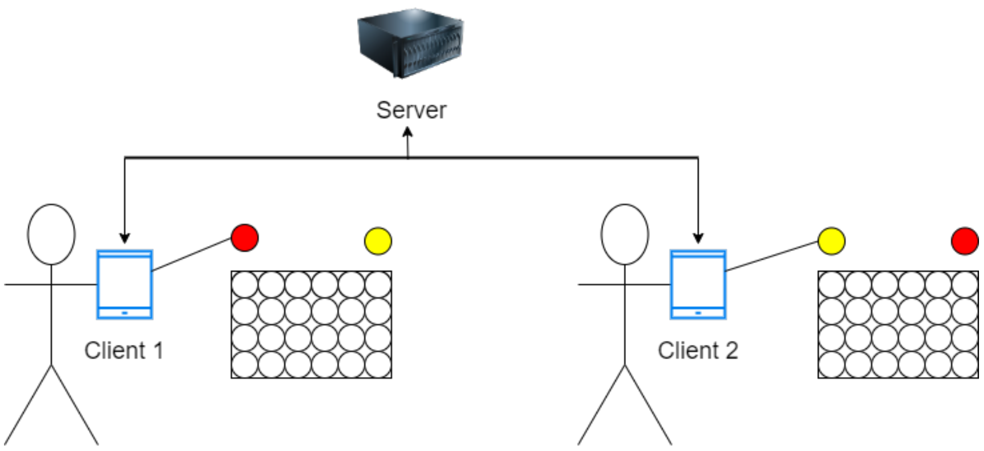


Figure 4.9: Distributed Architecture Marker-less

4.8 Application Workflow

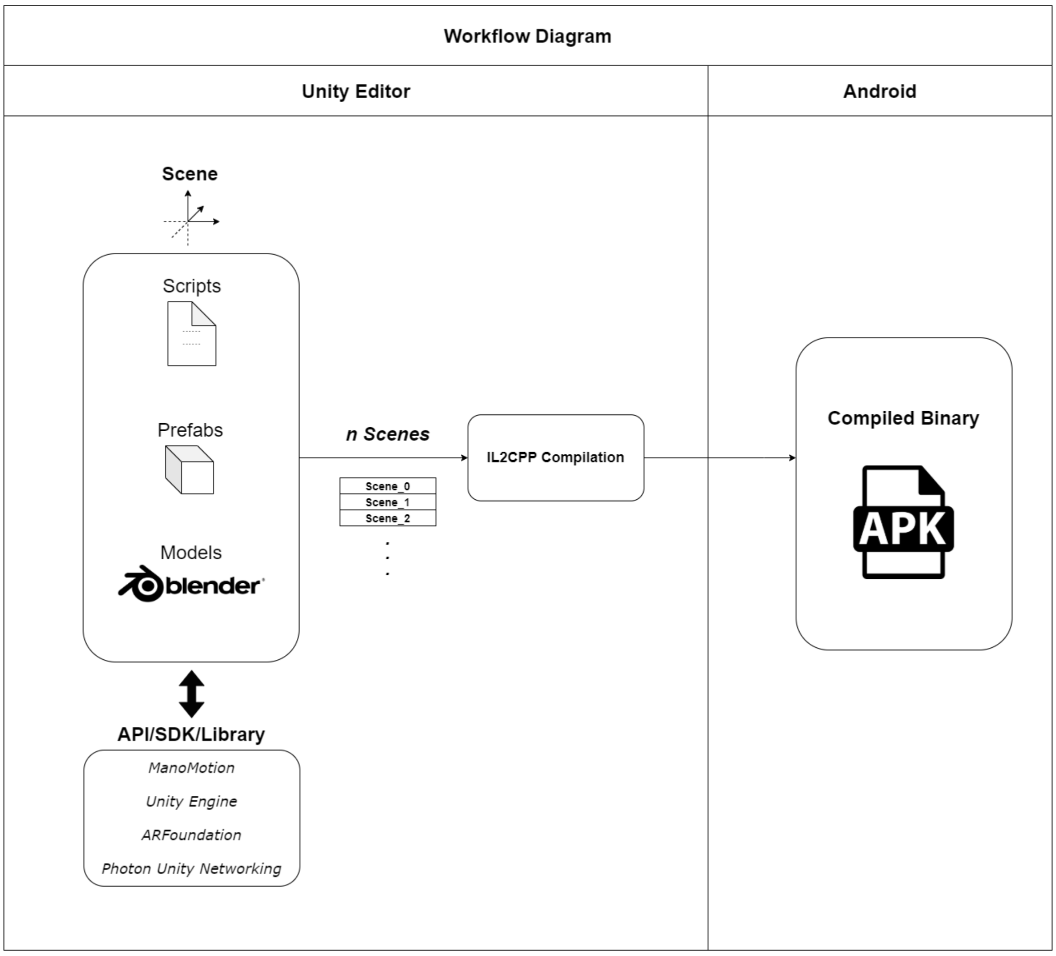


Figure 4.10: Application Workflow Diagram

4.9 Application Architecture

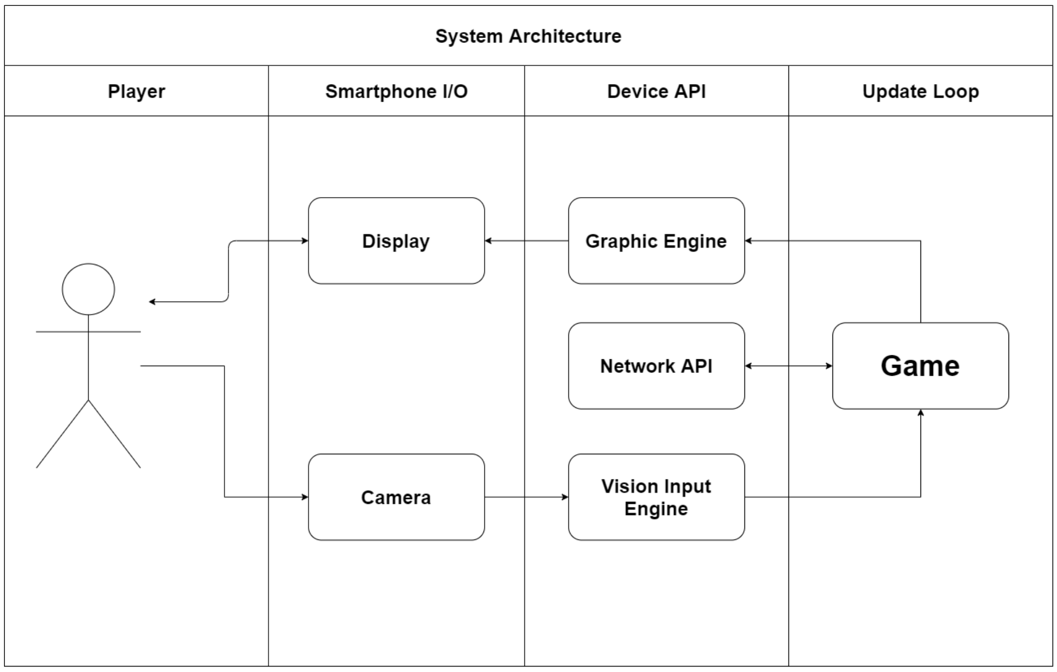


Figure 4.11: Application Architecture

4.10 Application Walkthrough

Home Scene

At the home scene, the user first enters an alias to use for the duration of the game. After an alias has been entered, they are then presented with a list of existing rooms available for them to join. Alternatively, they can begin their own game room and an opponent can subsequently join them. Once they have created or joined a room, they are presented with a loading screen followed, once it has loaded, by the gameplay scene.

Gameplay Scene

At the beginning of the gameplay scene, the user is first presented with an option to register the game board using a marker or without marker. If marker was selected the application will begin searching the images produced by the camera for the pre-determined marker and place the game board in the scene at the location of that marker. For the marker-less choice, the user is presented with planes found by the application, the board follows a cursor at the centre of the screen of the device and a place button is now visible which allows the player to place the game board in the environment. Once placed, the user can adjust it's position. When in a satisfactory position and 2 players are in the game room, each user can then select start game to spawn their pieces in the environment. Once pieces are in the environment, the players can compete to win the game.

4.11 Application Activity Diagram

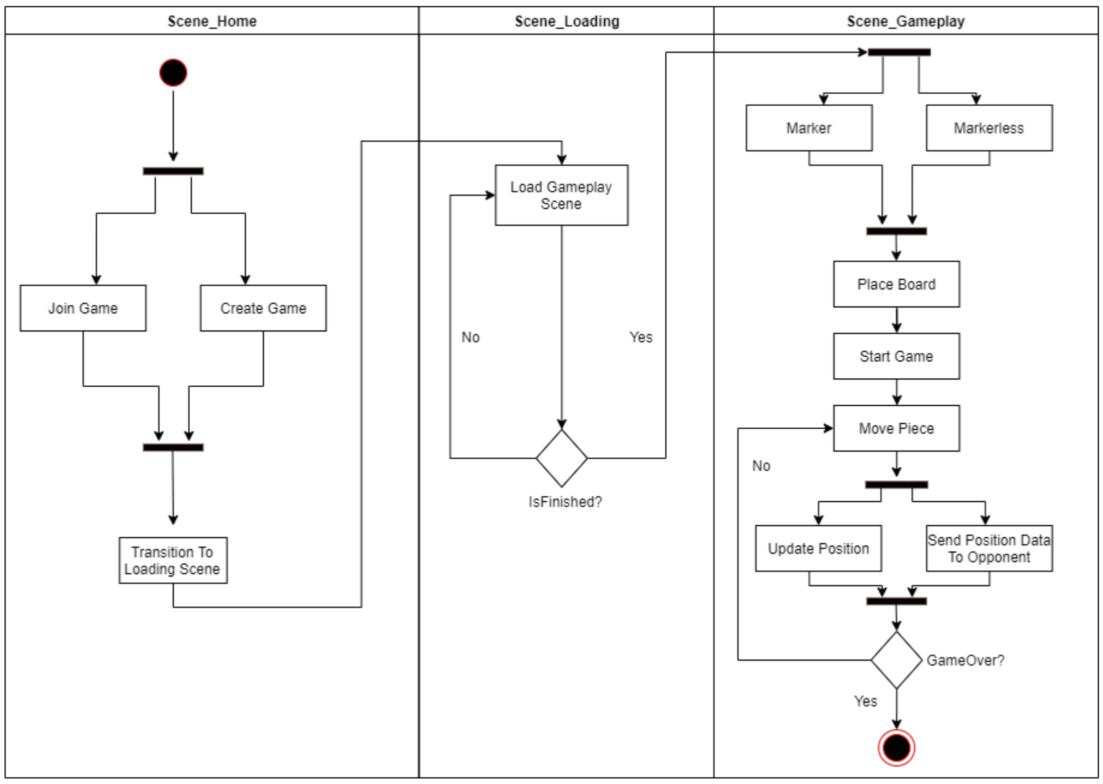


Figure 4.12: Game Activity Diagram

5 Interaction Implementation

The screen-based interactions were implemented using Unity's Input class and Physics class from the Unity scripting API [63]. The device used for this interaction is the multi-touch display of the smartphone, while the gesture interaction is implemented using a Software Development Kit by ManoMotion [44]. This SDK provides methods for using the camera of the device to detect gestures of a hand. As mentioned previously, hand gestures can be broadly categorised as either continuous or static gestures. Continuous gestures are recognised over many frames, while a static gesture is just recognised from a single frame. The SDK provides classes that can be used out-of-the-box for both of these gesture types, I will however only be using static gestures to recognise both grabbing and dropping of the pieces.

5.1 Raycast Interaction

The raycast interaction is implemented by continuously polling for a touch on the screen of the device during the fixed update loop of the game. If a touch is detected, first the co-ordinates of the location of the touch on the screen is recorded through the input class, then the physics class is used to raycast into the scene and, if the ray collides with a piece, the piece is selected. A selected piece will have its colour changed to green to indicate to the user that their action was successful as well as to bridge the evaluation gulf. This change to green is for both players, irrespective of the original colour of their pieces.

While a piece is selected and the user is touching the screen, the piece's position will be updated to be at a set distance along the ray until the user removes their finger from the screen. The piece will return to its un-selected state and experience the force caused by gravity. The piece will return to its original colour as well to indicate the interaction has ended.

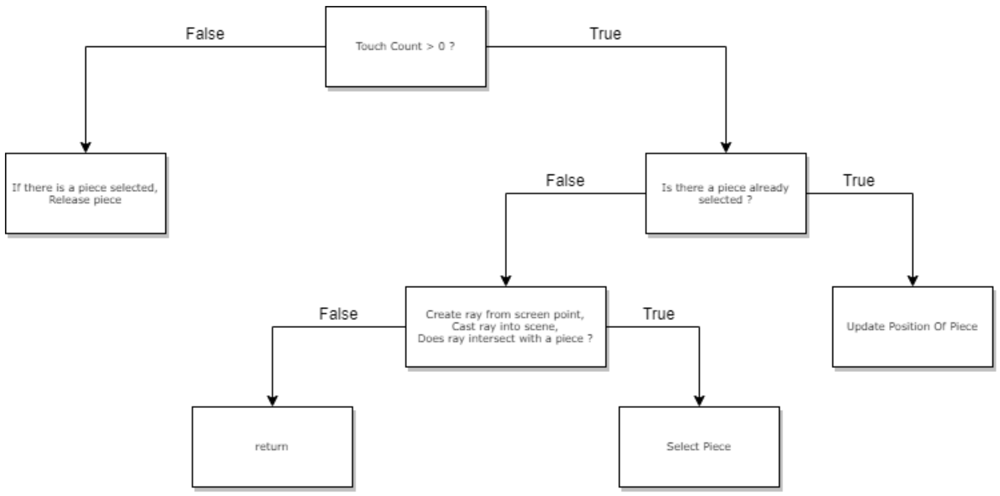

Touch Count > 0 ?
False
True
If there is a piece selected, Release piece
Is there a piece already selected ?
False
True
Create ray from screen point, Cast ray into scene, Does ray intersect with a piece ?
Update Position Of Piece
False
True
return
Select Piece

Figure 5.1: Raycast Interaction Logic

5.2 Pinching/Zoom Interaction

The pinching/zoom interaction is implemented similarly to the raycast interaction. It supplements this interaction by now allowing variable depth be added to the selected piece rather than it being locked to a constant distance. This interaction inherits all the functionality of the raycast interaction. The first instance a second touch is registered during the fixed update loop, the screen distance between the two touch positions is stored. During the next instance of the fixed update loop, if the distance between the two touch positions has increased, the distance of the piece from the camera is incremented proportional to the distance travelled. This new distance is subsequently stored and used during the next instance of the fixed update loop. The logic for moving/selecting the piece from Section 5.1 is subsequently inherited and used.

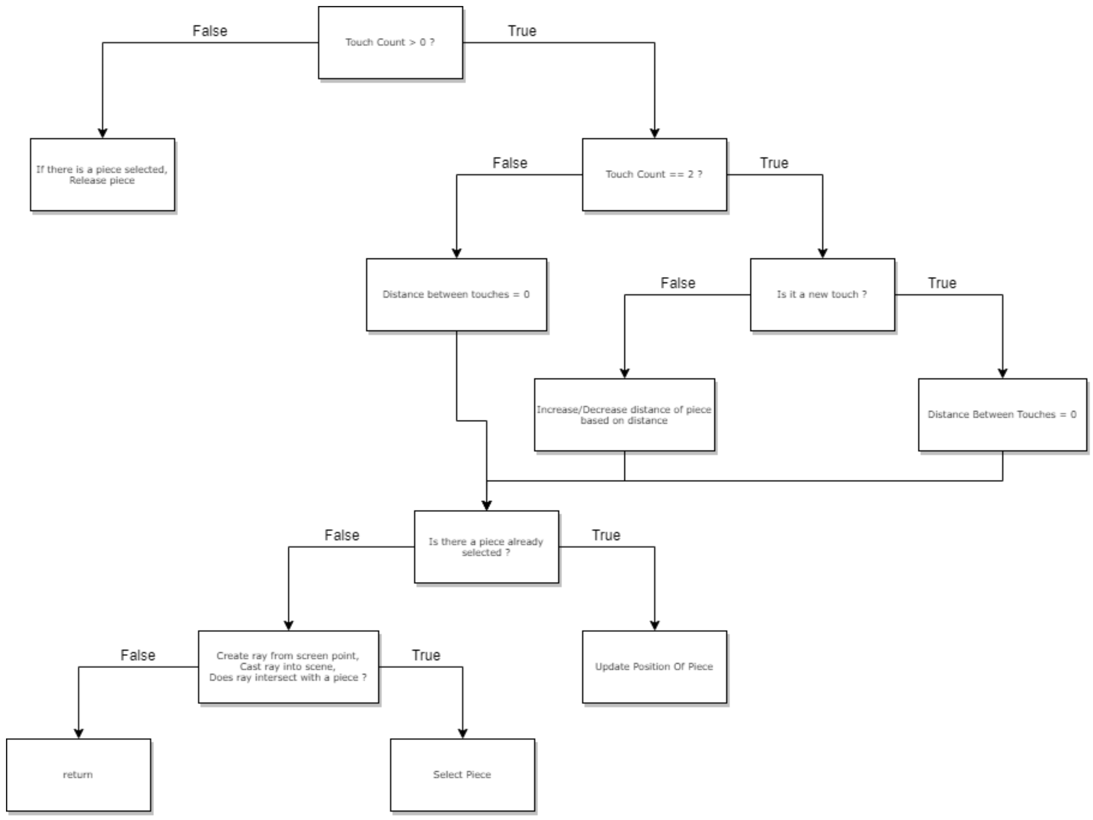

Touch Count > 0 ?
False
True
If there is a piece selected, Release piece
Touch Count == 2 ?
False
True
Distance between touches = 0
Is it a new touch ?
False
True
Increase/Decrease distance of piece based on distance
Distance Between Touches = 0
Is there a piece already selected ?
False
True
Create ray from screen point, Cast ray into scene, Does ray intersect with a piece ?
False
True
Update Position Of Piece
return
Select Piece

Figure 5.2: Pinch Interaction Logic

5.3 Gesture Interaction

The gesture interaction is implemented by first categorising the detected static gestures from the ManoMotion API into two categories: one for selecting a piece and one for de-selecting or dropping the piece. The static gestures recognised by the API are: click, grab, pick, drop and release. Click, grab and pick gestures were used for selection with the remaining being used to de-select the piece.

The API is then used to determine the detected position of the contact between the thumb and finger on the visible hand. A UI marker is displayed on the screen to indicate to the user where the API has determined this point of interest is. This is used to raycast into the scene similarly from the aforementioned 2-dimensional interactions. Adding the third dimension to this interaction type is achieved by estimating the depth of the hand from the camera. Depth information returned from the API is of type float, ranging from 0 to 1. Playtesting of the depth estimation value from the API saw that the value rarely dropped below 0.5, this was because as the hand got closer to the camera below this value, focus issues arose that meant that the hand could no longer be detected. An implementation decision made to overcome this limitation was to square the depth estimation value: this results in a higher range of possible values for the variable depth, now allowing for a greater range of values to be achieved. Figure 5.4 details the greater range of values for an exponential depth estimate compared to a linearly increasing depth estimate in Figure 5.3

Using the detected point of interest on the screen of the device and the estimated distance from the camera, a ray can be cast into the scene and a point along can then be used as a 3-dimensional cursor. A sphere model was used to indicate the point of interest between the index and thumb finger in the virtual environment. Intersecting of this model and a piece would indicate using the same highlighting mentioned previously that this piece is between your thumb and index and is able to be grabbed. Ambiguity surrounding multiple pieces possibly intersecting with the 3D cursor is overcome by allowing only one piece to be highlighted or selectable at a given time. The cursor must stop intersecting with the first piece to highlight a different piece.

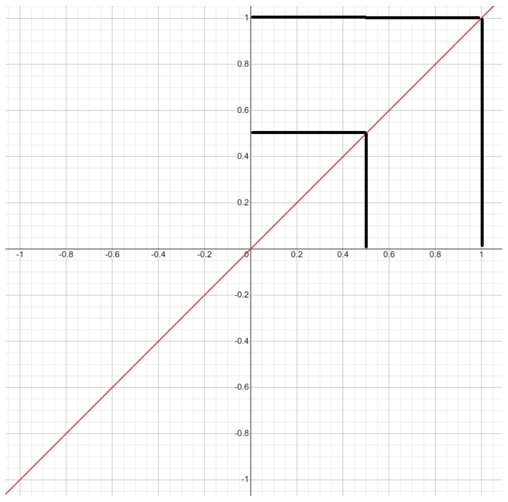


Figure 5.3: Linearly Increasing Depth

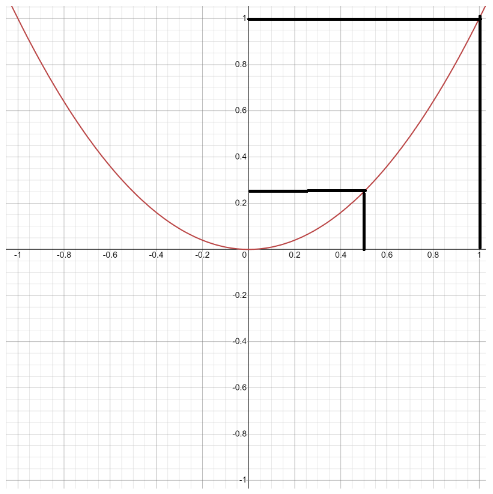


Figure 5.4: Exponentially Increasing Depth

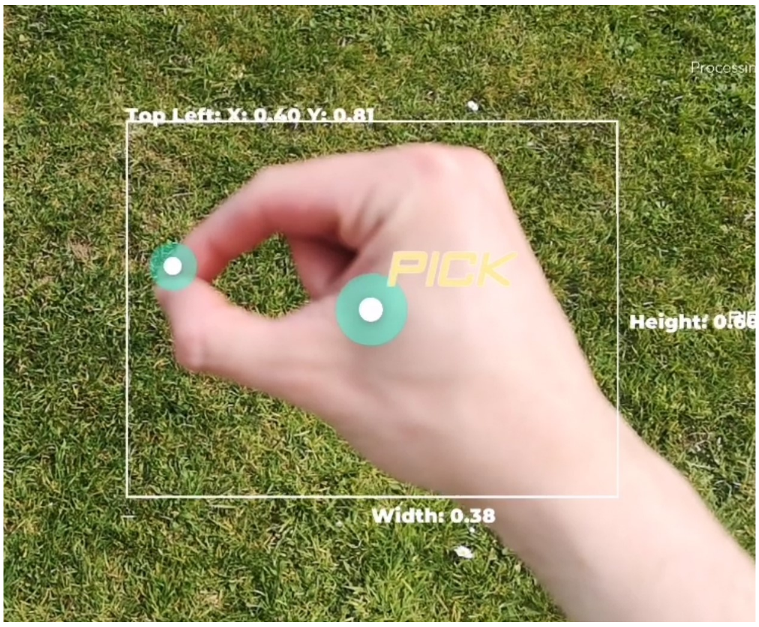


Figure 5.5: Detected Picking Gesture

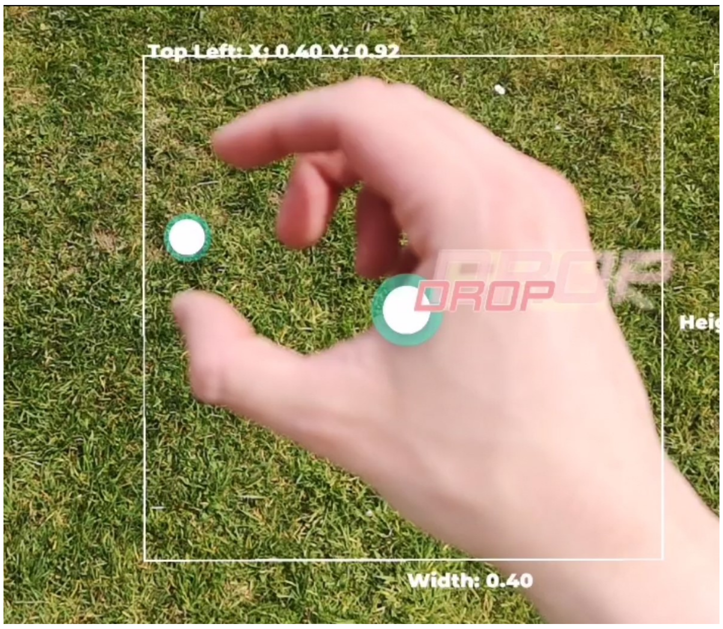


Figure 5.6: Detected Dropping Gesture

6 Results

6.1 Application Performance

The application implemented was tested on two AR-capable smartphone devices. The hardware specifications of each device can be found in table A.4 in the Appendix. Measured on both devices was the performance metric of frames per second of the application while using both the touchscreen interaction method and the gesture-based interaction method. Frame rate measures the number of times per second the graphics of the application are re-drawn. This can be calculated at each frame as the inverse of the time taken to draw the last frame.

$$\text{Let}\, T(f) = \text{The UNIX time when frame f is drawn.}$$

$$\text{Frames/Second at frame f} = \frac{1}{T(f) - T(f-1)}$$

$$\text{Set of random sample of 100 frames, } f_{100} = (fps_1, fps_2, ..., fps_{100})$$

$$\text{Average frame rate} = \frac{\sum(fps_1, fps_2, ..., fps_{100})}{100}$$

The time taken to draw the last frame can be retrieved from the UnityEngine scripting API [63]. The inverse of this number was logged to a text file for both devices. The data for these text files were processed before being visualised. A random sample of 100 frame logs were taken for both interaction methods. Graphs of these samples can be found in figures A.5 - A.8. Table shows the average frames per second for both interactions on both devices.

	Asus Zenfone AR ZS571KL	Huawei P20 Pro
Screen Interaction	58.66	48.34
Gesture Interaction	11.94	10.31

Table 6.1: Frames per second achieved

Using the gesture interaction incurs a frame rate loss of 80% and 79% for the Asus and Huawei devices respectively.

6.2 Error Rates of the Gesture Interaction

A controlled experiment was conducted to measure the impact of using this application in different environments. The controlled experiment gives results for both the gesture classification and tracking accuracy of the application. This experiment is conducted to measure the error rate of the interaction with different backgrounds of the camera's image. The experiment attempts fifty grabs/selections of a piece, direct manipulation of the piece followed by dropping/de-selection of the piece. The success rate of the pick gesture will be measured as a ratio of successes to failures for the fifty attempted interactions.

$$Pick = (x_1, x_2, ..., x_{50}), \text{ where } x_i = 1 \text{ (Success) or } 0 \text{ (Fail)}$$

$$Success(Pick) = \sum(x_1, x_2, ..., x_{50})$$

$$Failure(Pick) = 50 - Success(Pick)$$

While the success rate for the drop interaction will measure the same ratio of successes to failure except only for the successful pick interactions.

$$Drop = (x_1, x_2, ..., x_{Success(Pick)}), \text{ where } x_i = 1 \text{ (Success) or } 0 \text{ (Fail)}$$

$$Success(Drop) = \sum(x_1, x_2, ..., x_{Success(Pick)})$$

$$Failure(Drop) = Success(Pick) - Success(Drop)$$

Two distinct errors can occur from these interactions, either the gesture is miss-classified and therefore the user cannot directly manipulate the piece or the tracking location of the hand causes the 3D cursor to not intersect the desired piece and therefore the subject of the interaction is missed and the user must repeat the attempt. The experiment is conducted in three distinct environments to examine the impact it has on the ability to use this interaction, it was conducted: outside on a grassy surface, on a wooden table and on a white texture-less table.

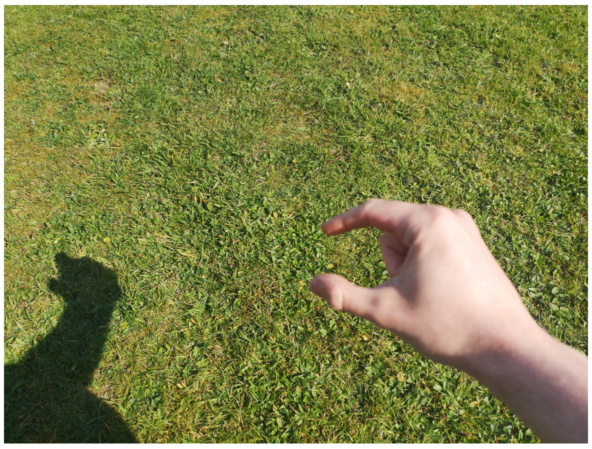


Figure 6.1: Experiment on Grass

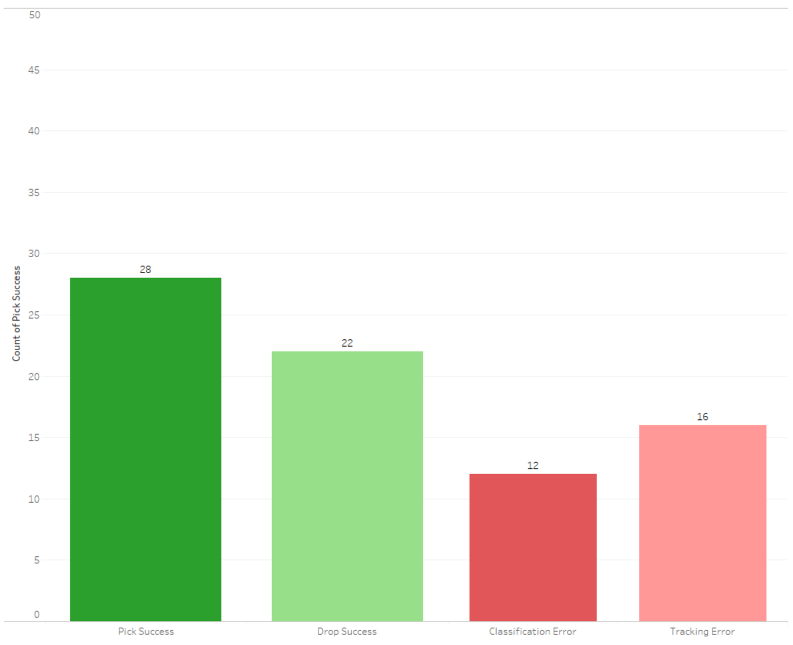


Figure 6.2: Results of Interaction Attempts on Grass

The first scenario for the interaction was outside on a grassy environment on a clear, sunny day. An image from the camera can be seen at Figure 6.1 and the results from this scenario are presented in Figure 6.2. 28 out of the 50 (56%) attempted pick gestures were successful, while 22 of the 28 (79%) of the attempted drop gestures were successful. Of the 28 errors, 12 (42%) of them were due to miss-classifying the interaction and 16 (58%) caused by errors tracking the hand.

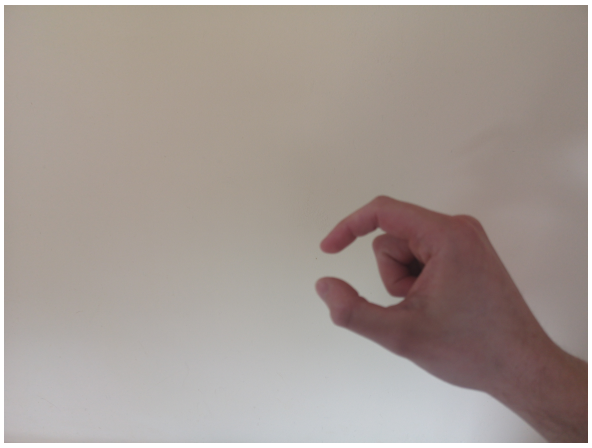


Figure 6.3: Experiment with white backdrop

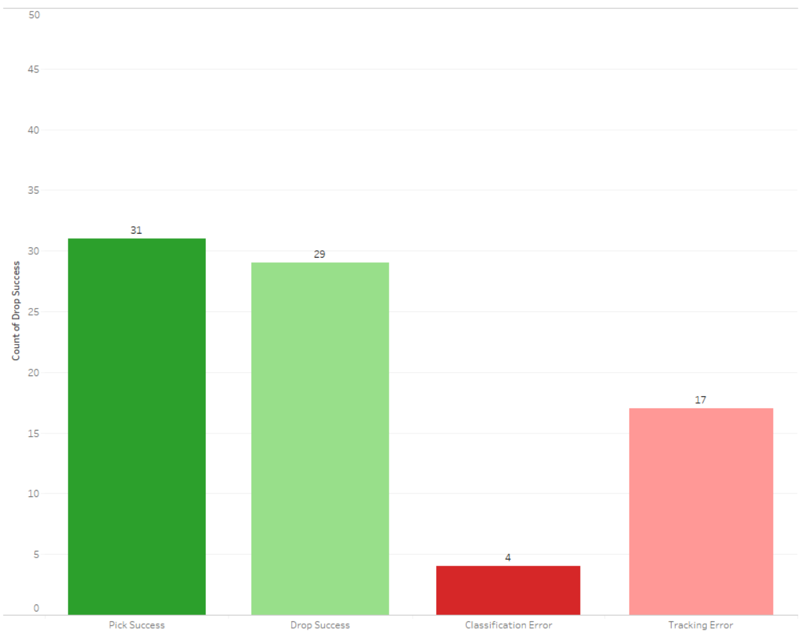


Figure 6.4: Results of Interaction Attempts with White Background

The second scenario for the interaction was on a white, texture-less table. This was conducted inside in a windowed, well-lit room during the afternoon. The adjacent wall is also of a similar white colour. An example image from the camera can be seen at Figure 6.3 and the results from this scenario are presented in Figure 6.4. 31 out of 50 (62%) pick interactions were successful, while 29 of 31 (94%) drop interactions were successful. Of the 21 errors that occurred, 4 (19%) of these were due to miss-classification of the gesture and 17 (81%) were due to incorrect tracking of the hand.

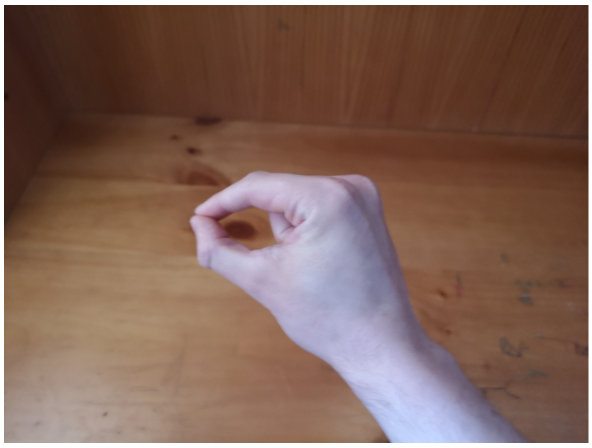


Figure 6.5: Experiment on Wooden Tabletop

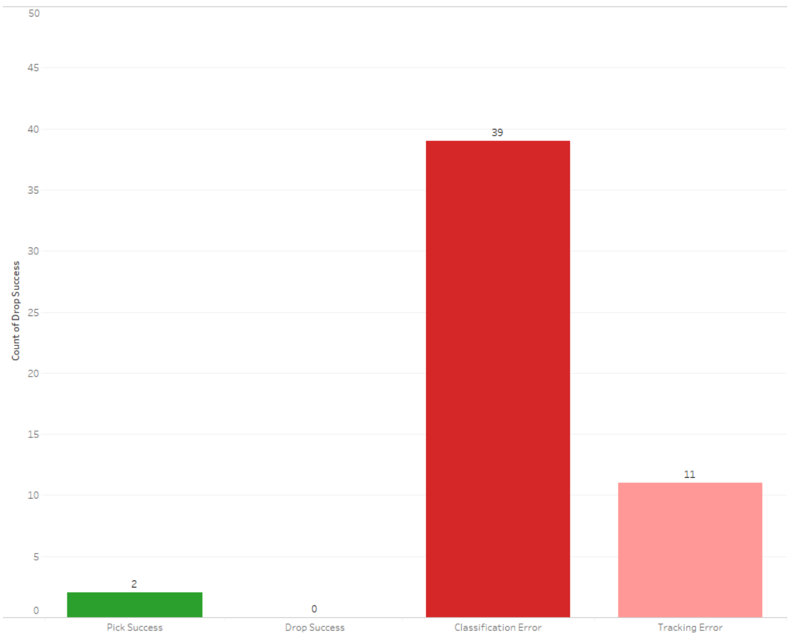


Figure 6.6: Results of Interaction Attempts on Wooden Tabletop

The final scenario occurred on a wooden tabletop, in an indoor setting that was well-lit with windows. An example image from the camera can be seen at Figure 6.5 and the results are visualised in Figure 6.6. This scenario resulted in the most amount of errors with zero successful attempts at achieving all of selection, manipulation and de-selection of a piece. Only 2 attempts of 50 (4%) were successful for picking/selecting the piece, none of the attempts to subsequently drop it were successful, and of the 50 total errors, 39 (78%) of attempts were classification errors and 11 (22%) were tracking errors.

7 Evaluation

The experiment itself gave a good variation of environments where the application performs well, moderately and poorly. The experiment on the white background proved to have the highest success rate. The lower amount of extraneous features extracted from the image resulted in almost all gestures classified. Errors from tracking the hand were the biggest offender in this scenario, these errors could be due to both inaccuracy from the user or the detection.

An improvement of my implementation to reduce the occurrence of tracking errors of interaction would be to set a "speed limit" of the 3-dimensional cursor. From observation of the tests, some tracking errors were caused due to the motion of the hand during a gesture. While the user receives feedback that they can interact with a piece, this interaction still occasionally fails as their hand movements incur a change to their tracked location. This causes the cursor to stop intersecting with the piece before the interaction has finished. Implementing a constant speed of the cursor would reduce the number of these incurred errors. The piece won't have enough time to stop intersecting the piece and their interaction will be successful. This wouldn't be suitable for applications where the cursor should be able to move as fast as a user's hand.

Another limitation observed throughout this study is the degree in which each trial of an experiment can be exactly replicated. There is an inherent random variability for independent trial of the interaction conducted in the experiment due to the impossibility of identical replication between each trial of the interaction. The experiment, therefore has an underlying assumption that all errors incurred were due to the limitations of the system and therefore our results also reflect some amount of human error.

8 Conclusion

This dissertation has detailed the design, implementation and experiments conducted to examine the capabilities of 3-dimensional interactions in video-based AR smartphone applications. The application allows two users to compete with each other at playing a real-time augmentation of a tabletop classic game. Multi-modal interactions were implemented to benchmark the impact of the gesture interaction.

The application behaves similarly on both devices with the frame rate taking a significant hit with the vision-based gesture interaction enabled. This drop is easily explained by the vision based gesture detection being called on each frame captured by the camera. It doesn't affect the game to the point where it cannot be played, but certainly has a noticeable latency for detecting the interaction. Assuming an average frame rate on all AR capable smartphone devices of 11 frames per second while using the gesture interaction, this will incur a latency of 91ms between performing the gesture and seeing the gesture occur on the smart phone screen.

UnityEngine executes all user defined scripts and behaviours in a single-thread. The vision library functions provided by the ManoMotion SDK are notably causing this bottleneck in the single-thread. This stage of execution is the application stage of the graphics pipeline and as Akenine-Moller notes, the slowest stage of the graphics pipeline will determine the frames per second the application achieves [30].

The dramatic difference in the errors incurred from the gesture interactions due to different environments also poses limitations for the variety of locations where the gesture mode of interaction can be used successfully. Wooden tabletops are feature rich which benefits marker-less registration of virtual objects on detected planes at the expense of almost entirely unusable gesture interactions. Meanwhile, featureless surfaces such as a white plane have high levels of success for the gesture interaction while marker-less registration of virtual objects with these images can prove difficult.

The experiment conducted with the wooden table resulted in a completely unplayable experience with zero fully successful select and de-select interactions. The textured features of the table resulted in the 3d cursor rarely ever tracking the users actual hand position. See figure in the Appendix. Gestures were also rarely correctly classified.

Finally, the experiment conducted on the grass surface resulted in a good experience with a moderate amount of accuracy. Just under half (43%) of the errors were caused by not classifying the gesture correctly.

9 Further Work

9.1 User Study to Evaluate Interaction Methods

A user study could be conducted to determine if the game has a greater satisfaction when played by two co-located competing players. Also this study could be used to gain an insight into how much the frame rates affect the satisfaction of the game. A similar study on a first person shooter game was conducted by Claypool et al. [17]

The Hypotheses to test.

- *The gesture interaction method becomes more satisfactory in a collaborative environment where two players can see each others gestures.*
- *The low frame rates impacted the satisfaction of the gesture based interaction.*

A quantitative study like this would require at least 20 people according to Nielsen [49] and Holzinger [34] proposes using 30 people for gathering this type of data. A time of 3 minutes was used by Billinghurst et al. [33] and 5 minutes was used by [43]. 5 minutes would be appropriate for this user study.

A questionnaire with 4 statements using a Likert Scale of five responses ranging from strongly disagree (1) to strongly agree (5) could be used to evaluate the participants opinion on the usability of this

A Questionnaire for discovering usability issues for each interaction method.

- *There was a noticeably low amount of frames rendered by the application which reduced the satisfaction of playing.*
- *This interaction had greater satisfaction/engagement when playing with a visible opponent.*

9.2 Optimising the Application Stage

The poor frames achieved by the application using the gesture interaction was largely due to the vision library being added to the update loop in UnityEngine. As mentioned previously, user defined scripting is performed on a single thread. Further investigation into using a multi-threaded approach to parallelise the gesture interactions from other scripts in the update loop could reduce the incurred latency. Studies have shown frame rates affect both a users performance in video games as well as their perception of the quality of the picture [17].

Bibliography

[1] *ISO 9241-11:2018(en) Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts.*

[2] *Mono Runtime.*

[3] K. Abhishek, L. Qubeley, and D. Ho. Glove-based hand gesture recognition sign language translator using capacitive touch sensor. pages 334–337, 08 2016.

[4] L. Ahmed, D. Hegazy, S. Hamdy, and T. Elarif. Marker-based finger gesture interaction in mobile augmented reality. pages 115 – 133, 2019.

[5] T. T. Ahonen. Ok, am willing to call it: Ar is the 8th mass medium (augmented reality). 2012.

[6] Apple. Understanding arkit tracking and detection.

[7] R. T. Azuma. A survey of augmented reality. *Presence: Teleoper. Virtual Environ.*, 6(4):355–385, Aug. 1997.

[8] H. Bai, G. Lee, and M. Billinghurst. Freeze view touch and finger gesture based interaction methods for handheld augmented reality interfaces. pages 126–131, 11 2012.

[9] M. Bajura, H. Fuchs, and R. Ohbuchi. Merging virtual objects with the real world: Seeing ultrasound imagery within the patient. *SIGGRAPH Comput. Graph.*, 26(2):203–210, July 1992.

[10] T. Baudel and M. Beaudouin-lafon. Charade: Remote control of objects using freehand gestures. 04 1994.

[11] R. Bell. *Board and table games from many civilizations.* Dover Publications, 1 1979.

[12] M. Billinghurst, A. Cheok, S. Prince, and H. Kato. Real world teleconferencing. *IEEE Computer Graphics and Applications*, 22:11–13, 01 2002.

[13] M. Billinghurst and H. Kato. Collaborative augmented reality. *Communications of the ACM*, 45, 01 2003.

[14] C. Booth. Alan kay and the graphical user interface.

[15] O. R. Chanu, A. Pillai, S. Sinha, and P. Das. Comparative study for vision based and data based hand gesture recognition technique. In *2017 International Conference on Intelligent Communication and Computational Techniques (ICCT)*, pages 26–31, 2017.

[16] J. Choi, H. Park, J. Park, and J.-I. Park. Bare-hand-based augmented reality interface on mobile phone. pages 275–276, 10 2011.

[17] M. Claypool, K. Claypool, and F. Damaa. The effects of frame rate and resolution on users playing first person shooter games. *Proceedings of SPIE - The International Society for Optical Engineering*, 6071, 01 2006.

[18] R. Cutler and M. Turk. View-based interpretation of real-time optical flow for gesture recognition. 11 1997.

[19] H. Desurvire, Heather, Caplan, Martin, Toth, and J. A. Using heuristics to evaluate the playability of games. 01 2004.

[20] H. Desurvire and C. Wiberg. Game usability heuristics (play) for evaluating and designing better games: The next iteration. pages 557–566, 07 2009.

[21] A. Duenser, Rapha, H. Seichter, and M. Billinghurst. Applying hci principles to ar systems design. 01 2007.

[22] D. Dykstra. A comparison of heuristic evaluation and usability testing: The efficacy of a domain-specific heuristic checklist. 1993.

[23] M. Elmahgiubi, M. Ennajar, N. Drawil, and M. Elbuni. Sign language translator and gesture recognition. 06 2015.

[24] S. Feiner, B. Macintyre, T. Höllerer, and A. Webster. A touring machine: Prototyping 3d mobile augmented reality systems for exploring the urban environment. volume 1, pages 74–81, 12 1997.

[25] B. Foundation. *About Blender*, 2020.

[26] W. T. Freeman, K.-I. Tanaka, K. Kyuma, and J. Ohta. Computer vision for computer games. pages 100–105, 10 1996.

[27] R. Freitas and P. Campos. Smart: A system of augmented reality for teaching 2nd grade students. pages 27–30, 01 2008.

[28] E. Games. *Photon Unity Networking*, 2020.

[29] N. N. Group. Usability 101: Introduction to usability. 2012.

[30] E. Haines, N. Hoffman, and A.-M. Tomas. *Real-Time Rendering, 3rd Edition.* Taylor Francis Group, 2008.

[31] I. Harris. Pokemon go captures 800 million downloads. 2018.

[32] N. Hedley. Empirical evidence for advanced geographic visualization interface use. pages 10–16, 09 2003.

[33] A. Henrysson, M. Billinghurst, and M. Ollila. Face to face collaborative ar on mobile phones. volume 2005, 01 2005.

[34] A. Holzinger. Usability engineering methods for software developers. *Commun. ACM*, 48:71–74, 01 2005.

[35] A. Hsu and C.-C. J. Kuo. On the design of multiplayer online video game systems. *Proceedings of SPIE - The International Society for Optical Engineering*, 11 2003.

[36] J. Huizinga. *Homo Ludens.* Random House, 1938.

[37] T. Höllerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway. Exploring mars: Developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers and Graphics*, 23:779–785, 1999.

[38] W. Hürst and C. Wezel. Gesture-based interaction via finger tracking for mobile augmented reality. *Multimedia Tools and Applications*, 62, 01 2012.

[39] N. Isaac. *Philosophiæ Naturalis Principia Mathematica.* 7 1687.

[40] K. Kiyokawa, H. Takemura, and N. Yokoya. Seamlessdesign for 3d object creation. *IEEE MultiMedia*, 7:22–33, 02 2000.

[41] S. M. Ko, W. Chang, and Y. G. Ji. Usability principles for augmented reality applications in a smartphone environment. *International Journal of Human-Computer Interaction*, 29, 08 2013.

[42] B. N. Laboratory. The first video game?

[43] R. Mandryk and K. Inkpen. Physiological indicators for the evaluation of co-located collaborative play. *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, pages 102–111, 01 2004.

[44] ManoMotion. Manomotion, 2020.

[45] E. Molla and V. Lepetit. Augmented reality for board games. pages 253 – 254, 11 2010.

[46] G. Murthy and R. Jadon. A review of vision based hand gesture recognition. *International Journal of Information Technology and Knowledge Management*, 2:405–410, 08 0002.

[47] J. New, E. Hasanbelliu, and M. Aguilar. Facilitating user interaction with complex systems via hand gesture recognition. 01 2003.

[48] J. Nielsen. Enhancing the explanatory power of usability heuristics. page 152–158, 1994.

[49] J. Nielsen. Quantitative studies: How many users to test? 2006.

[50] W. Piekarski and B. Thomas. Arquake: the outdoor augmented reality gaming system. pages 36 – 38, 2002.

[51] Pinelle, David, N. Wong, and Nelson. Heuristic evaluation for games: Usability principles for video game design. *Conference on Human Factors in Computing Systems - Proceedings*, 01 2008.

[52] W. Quesenbery. Balancing the 5es: Usability. *Cutter IT Journal*, 17:4–11, 02 2004.

[53] Y. Rogers, H. Sharp, and J. Preece. *Interaction Design, 4th edition.* Wiley, 2015.

[54] B.-K. Seo, J. Choi, J.-H. Han, H. Park, and J.-I. Park. One-handed interaction with augmented virtual objects on mobile devices. 12 2008.

[55] B. Shelton and N. Hedley. Using augmented reality for teaching earth-sun relationships to undergraduate geography students. page 8 pp., 02 2002.

[56] B. Stenger, P. Mendonca, and R. Cipolla. Model-based 3d tracking of an articulated hand. volume 2, pages II–310, 02 2001.

[57] I. E. Sutherland. The ultimate display. pages 506–508, 1965.

[58] I. E. Sutherland. A head-mounted three dimensional display. page 757–764, 1968.

[59] Z. Szalavári, D. Schmalstieg, A. Fuhrmann, and M. Gervautz. “studierstube”: An environment for collaboration in augmented reality. *Virtual Real.*, 3(1):37–48, Mar. 1998.

[60] Z. Szalavári and M. Gervautz. The personal interaction panel – a two-handed interface for augmented reality. *Computer Graphics Forum*, 16(3):C335–C346, 1997.

[61] D.-N. Ta, K. Raveendran, Y. Xu, K. Spreen, and B. Macintyre. Art of defense: A collaborative handheld augmented reality board game. 08 2009.

[62] U. Technologies. *ARFoundation*, 2020.

[63] U. Technologies. *Unity - Scripting API*, 2020.

[64] Ted. Unveiling the genius of multi-touch interface design | jeff han.

[65] F. Van Diggelen and P. Enge. The world's first gps mooc and worldwide lab using smartphones. 2015.

[66] D. Wagner, T. Pintaric, F. Ledermann, and D. Schmalstieg. Towards massively multi-user augmented reality on handheld devices. 3468:208–219, 05 2005.

[67] D. Wagner and D. Schmalstieg. First steps towards handheld augmented reality. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, ISWC '03, page 127, USA, 2003. IEEE Computer Society.

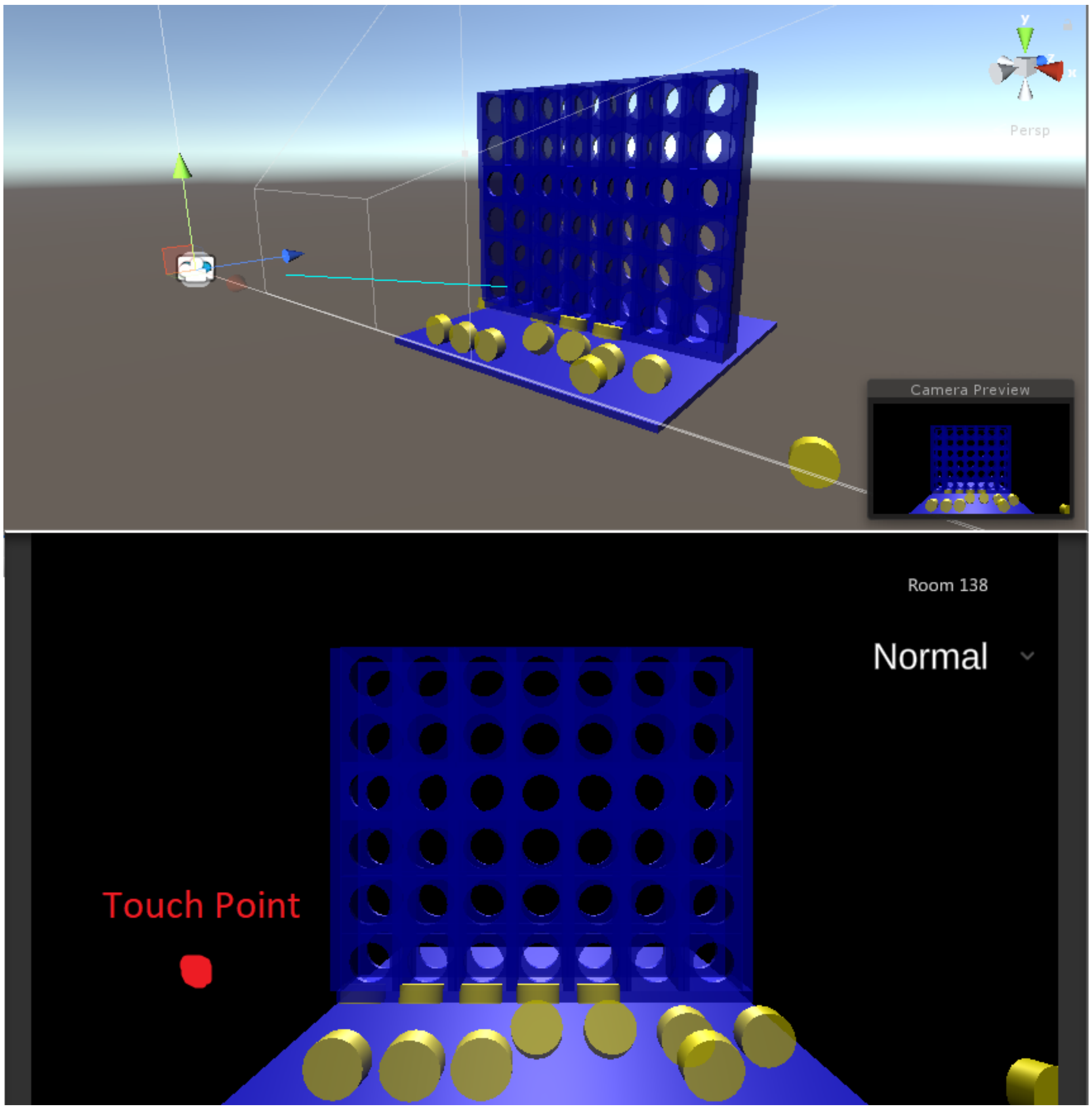
[68] X. Wang, M. Xia, H. Cai, Y. Gao, and C. Cattani. Hidden-markov-models-based dynamic hand gesture recognition. *Mathematical Problems in Engineering*, 2012, 04 2012.

[69] T. G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill. A hand gesture interface device. In *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface*, CHI '87, page 189–192, New York, NY, USA, 1986. Association for Computing Machinery.

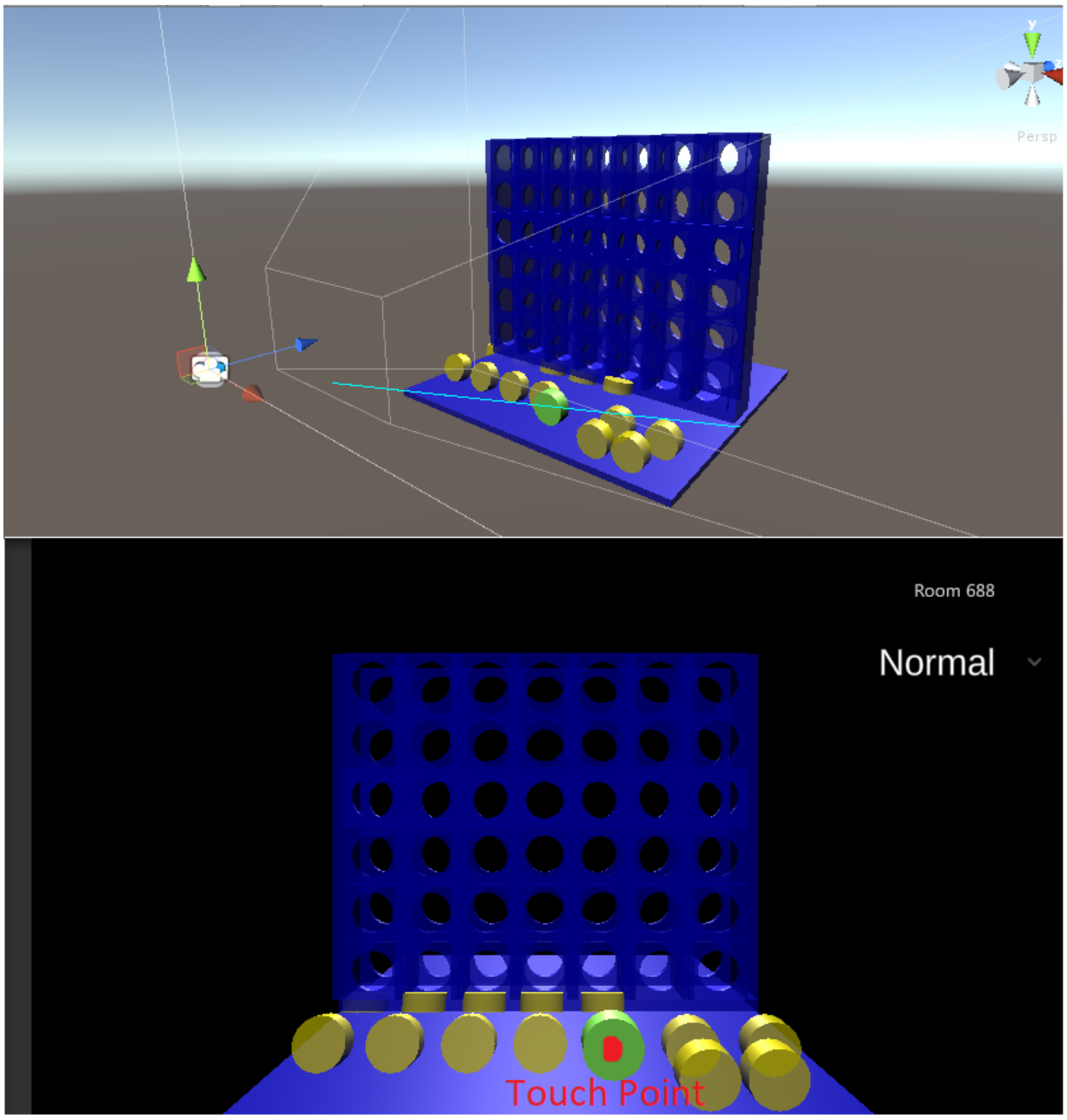
A.1 Github Repo Link

https://github.com/caolanb10/Connect_4

A.2 Raycast Miss



A.3 Raycast Hit



A.4 Smartphone Hardware Specifications

Hardware Specifications		
	Asus Zenfone AR ZS571KL	Huawei P20 Pro
RAM	6GB LPDDR4	6GB LPDDR4
Processor	Quad-Core @ 2.35Ghz	Octa-Core(4x2.4Ghz, 4x1.8Ghz)
Camera Rear	23MP Camera	40MP Camera
GPU	Adreno 530	Mali-G72 MP12

A.5 ASUS Smartphone Touchscreen Interaction

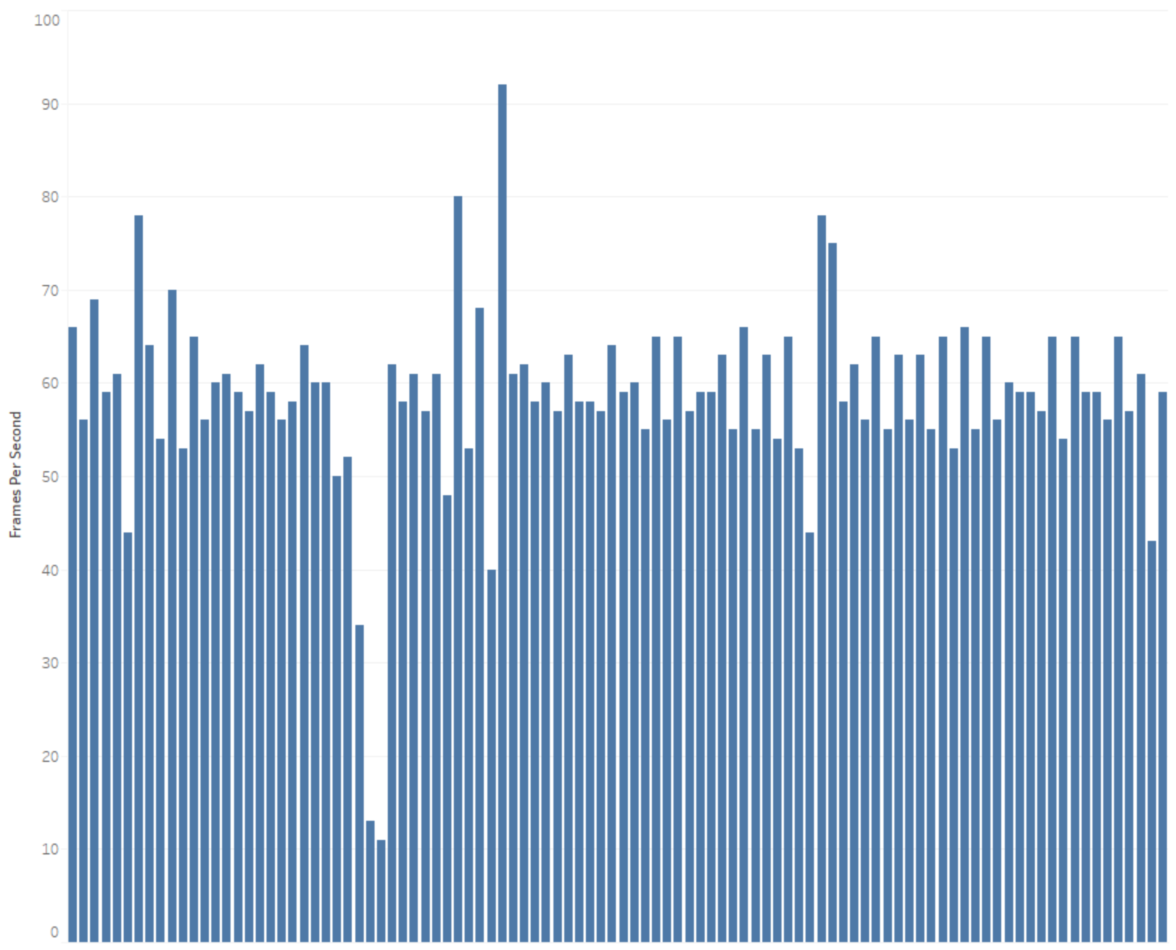


Figure 1: Frames Per Second ASUS Smartphone Touchscreen Interaction

A.6 ASUS Smartphone Gesture Interaction

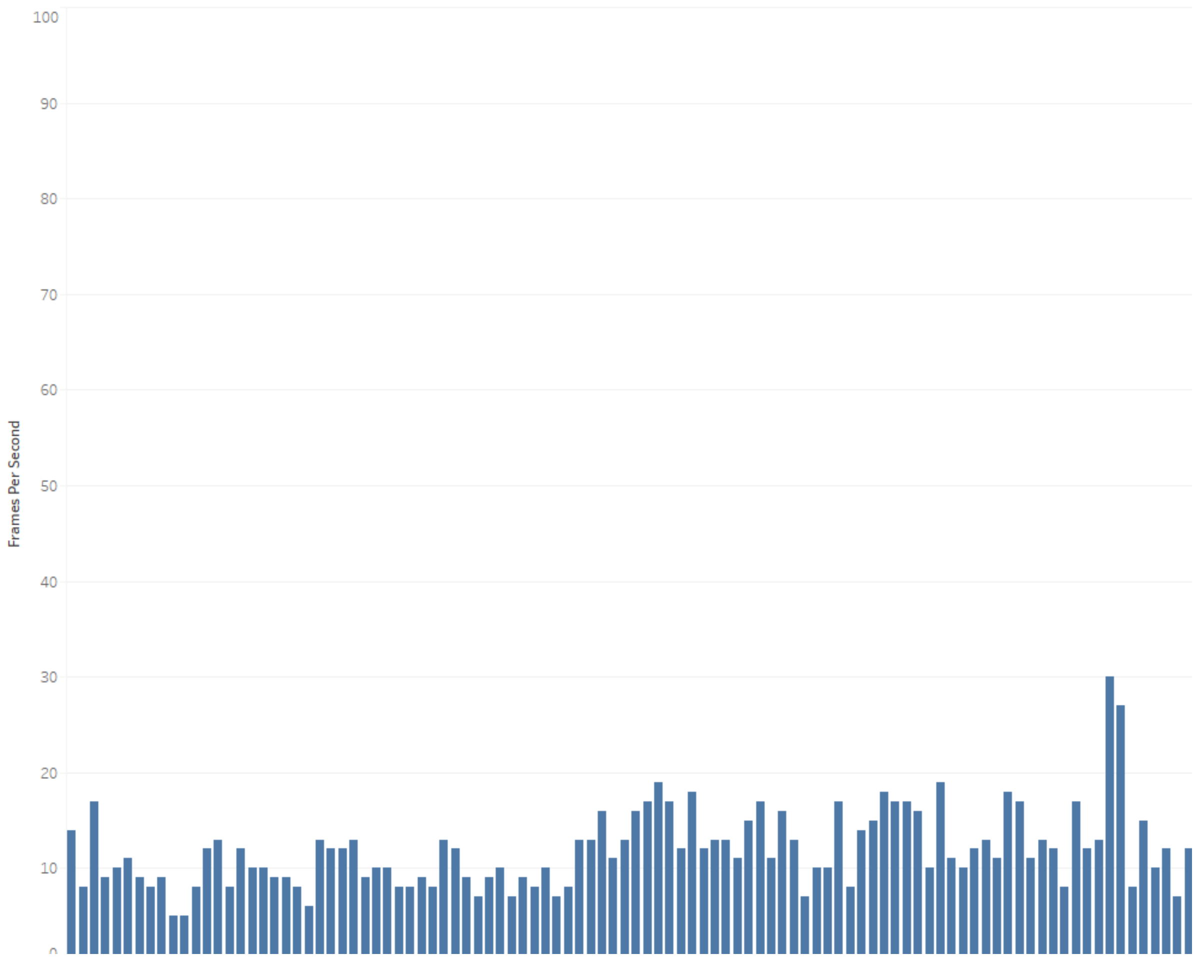


Figure 2: Frames Per Second ASUS Smartphone Gesture Interaction

A.7 Huawei Smartphone Touchscreen Interaction

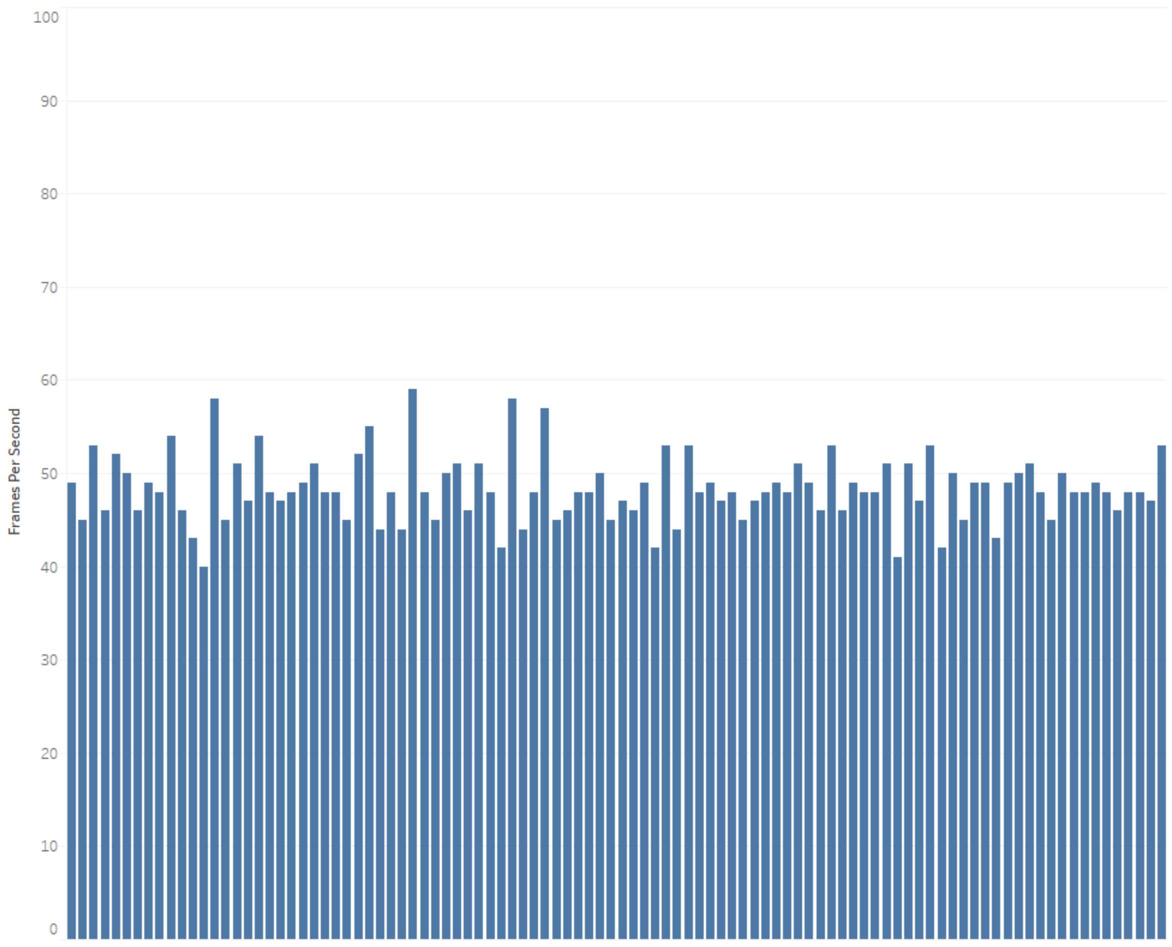


Figure 3: Frames Per Second Huawei Smartphone Touchscreen Interaction

A.8 Huawei Smartphone Gesture Interaction

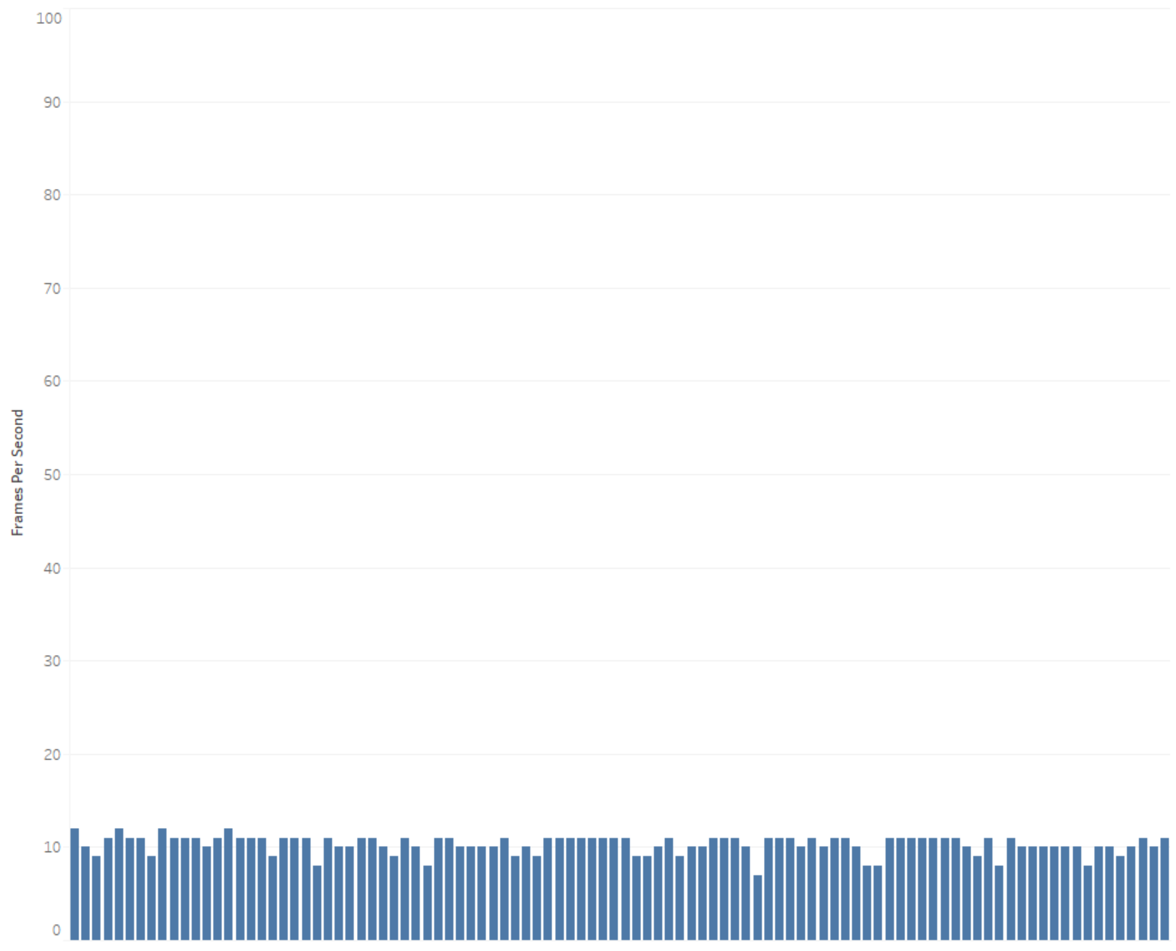
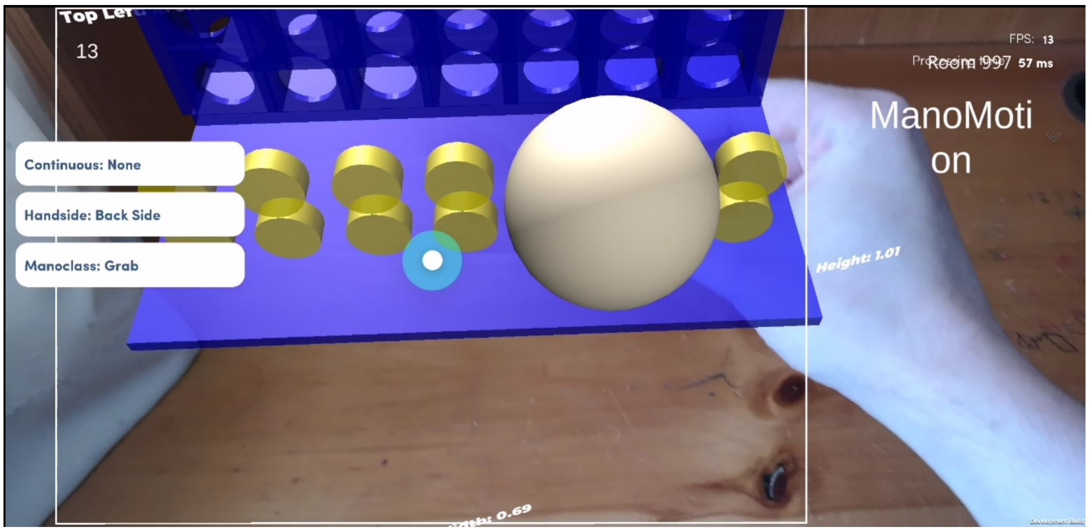


Figure 4: Frames Per Second Huawei Smartphone Gesture Interaction

A.9 Scenario: Wooden Table



Continuous: None
Handside: Back Side
Manoclass: Grab

Figure 5: Hand Position Tracking for Wooden Table

A.10 Scenario: Outside Scenario

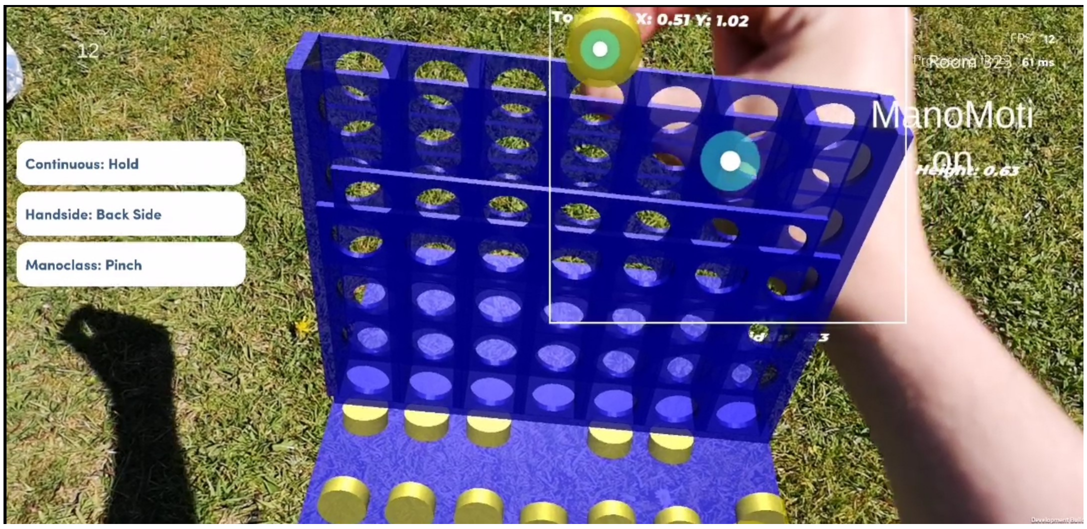


Figure 6: Hand Position Tracking for Outside Environment

A.11 Scenario: White Backdrop

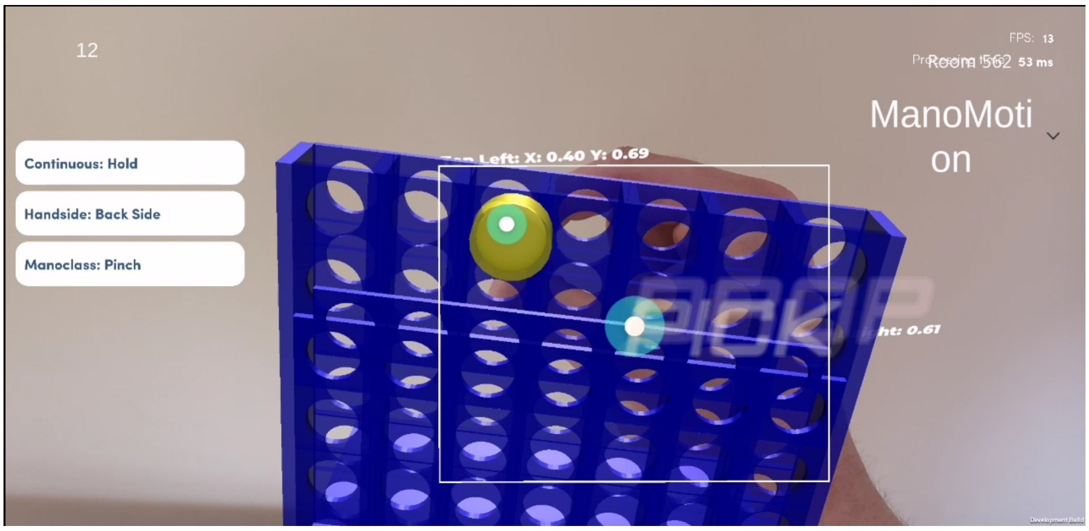


Figure 7: Hand Position Tracking for White Backdrop