TEL AVIV UNIVERSITY
School of Electrical Engineering

Game of Drones: "Pong" like Drone augmented reality game

A project submitted toward the degree of
Master of Science in Electrical and Electronic Engineering

by

Guy Hershtig

This project was carried out in The School of Electrical Engineering
Under the supervision of Mr. Yonatan Mandel and Prof. Amit Bermano

April 2020

Contents

Abstract

Project goal was to develop a drone-based Augmented Reality (AR) game.

The game will consist of two drones, and two mobile devices controlling the drones (e.g. tablets or cellphones). When viewing the drones through the devices, an AR augmentation changes their appearance, and adds the game content to the surrounding.

Game complexity starts with Pong and can evolve up to racing and dog-fighting. In collaboration with the TAU School for Computer Science, and the Game Technology Center – ETH Zurich.

Project stages decided to be:
- Literature overview
- Learning Unity, Vuforia, C#
- Learning drone communication
- Develop AR "Pong" like game
- Add drones to AR game
- Optional: collision avoidance system

Table of figures:

1 Introduction

1.1 Augmented Reality (AR), Virtual Reality (VR)

Augmented reality is a term used to identify a set of technologies that allows the view of real-world environment to be “augmented” by computer-generated elements or objects (Van Krevelen & Poelman, 2010[1]). More specifically, AR describes a mediated reality, where the visual perception of the physical real-world environment is enhanced by means of computing devices. Compared to Virtual Reality (VR), i.e. a set of technologies that allow the user to interact with a computer in a simulated environment (Khan et al. 2011[2]), AR does not aim to replace the real world with a simulated one and is consequently often classified as a mixed reality (MR) system. MR is a mix of reality and virtual reality, encompassing both AR and VR, via immersive technology (Milgram & Kishino 1994[3]).

The first AR prototypes, created by computer graphics pioneer Ivan Sutherland and his students at Harvard University and the University of Utah, appeared in the late 1960s and exploited a see-through display fitted to a helmet to present 3D graphics (Tamura 2002[4]).

From a technical point of view, since the late 1990s it has become much easier to develop AR applications rapidly, thanks to freely available software toolkits (e.g. the ARToolKit) and the development of camera systems able to analyze the physical environment in real time and relate positions between objects and environment in that period. This type of camera system now represents the basis for the integration of virtual objects with reality in AR systems. The smart phone adoption and the proliferation of AR browser applications are further aspects that contribute to the diffusion of AR solutions (Johnson et al. 2011[5]).

As a result, today AR solutions are increasingly diffuse in industrial contexts, where they are developed for different ends. Personal information or personal assistance applications have been developed on the basis of AR systems (Höllerer & Feiner 2004[6]). Other applications are reported in the tourism sector, military (Van Krevelen & Poelman 2010[7]; Henderson & Feiner 2009[8]), automotive & aerospace (Regenbrecht et al. 2005[9]) and medical sectors (Vogt et al. 2006[10]).

Another sector that is increasingly using AR/VR, is the gaming sector. Indoor and outdoor games are now utilizing cell phones for localization, multiplayer communication and for creating immersive games (Indoor game[11], Car racing game[12], PacMan game[13], Bowling[14]). Some of the games are meant for one player, while others are for two or more players ([15,16])

1.2 Drones

The word “drone” has referred to a male honeybee whose only role is to mate with the queen since Old English and it began branching out as a verb, meaning to buzz like a bee. Drone is also known as an unmanned aerial vehicle (UAV), an aircraft without a human pilot aboard and is often controlled by either a human operator or autonomously by onboard computers. So that, in computer science and artificial intelligence studies, they often call a drone as a remotely piloted vehicle (RPV), remotely operated aircraft (ROA), remote control helicopter (RC-Helicopter), or unmanned vehicle systems (UVS) (Eisenbeiss, 2004[17]).

society has begun to become familiar with a new drone technology, and many people started using drones for their own personal purposes resulting in using drones not only for military purpose, but also recreational and commercial purposes.

The largest theatrical Canadian entertainment company, Cirque du Soleil, designed a flying lamp show, called “SPARK”, with Swiss Federal Institute of Technology and ETH Zürich in 2014 (SPARKED: A Live Interaction Between Humans and Quadcopters, 2014[18]). They used multiple drones that are hidden inside each of the lampshades with a performer to show a live interaction between humans and flying robots.

Parrot, a French company introduced a first popular recreational quadcopter called, AR Drone shown at the CES 2010 Trade Show held in Las Vegas. The user can control the drone using their mobile devices such as a smart phone via an onboard Wi-Fi network. The drone is equipped with two cameras, one on the front and the other on the bottom and transmits live video to the user’s device over the wireless network.

Since then, many new cheap drones were introduced to market, enabling more people buying and using them, for recreational purposes as well as gaming (Air hogs connect[19], Drone racing game[20]).

In order to identify the drone and drone position, several techniques can be used. One of them is by using visual markers[21]. This technique requires good lighting in order to find markers that differ from their surroundings, and can’t always be used, especially outside the controlled environment of a lab preset, or if the markers are required to be mounted on the drone, which is problematic for very small drones, due to stabilization issues and the amount of weight the drone can carry.

Another technique to identify the drone position is by using Simultaneous Localization and Mapping (SLAM[22-25]), in which, a stream of images taken by the drone is used to mapping the drone surroundings and find the drone position in the map that was generated. SLAM can be performed with one or more cameras, but sending the video stream from the drone to a processing unit takes its toll on the drones power, and is impractical for very small drones, for which their battery allows few minutes for flight when not sending video stream, and even shorter flight time when it is used.

Other drone sensors, such as Inertial Measurement Unit (IMU), can be used to improve SLAM or to infer drone position, as was demonstrated in several papers[26-28].
DJI drones, for example, have height sensor on their bottom side, which enable to detect their height in about 10cm resolution. They also have image sensor on their bottom side to decrease drone drift while the drone is above the “Drone Pad”.

2 Methods

In this chapter, all of methods and project building blocks are described.

2.1 Quick start guide

Following this section is the fastest and easiest way to start playing.

The graphical user interface (GUI) comprises mostly of buttons. Not all of the buttons are presented at the same time, since cell phone area is small, and once a button is pressed, the required buttons appear while those who are no longer required are hidden.

The first menu of the game requires a selection between 4 main buttons:

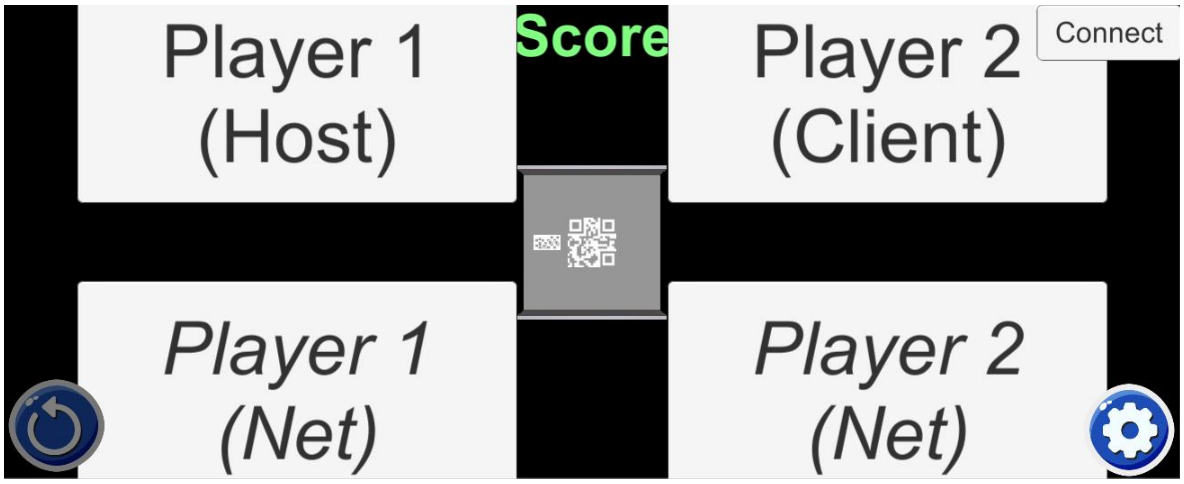


Figure 1: game from top view (before main QR detection)

Player 1 (Host) – be the host in a wi-fi game
Player 2 (Client) – be the client in a wi-fi game
Player 1 (Net) – be the host in an internet game
Player 2 (Net) – be the client in an internet game

The host and client control player1 and player2, which are identical in most aspects. The game is running both on the host and on the client, but the ball position is calculated only by the host and the position is sent to the client for rendering. Also, only the host can pause and resume the game, and the game starts once the host presses the play button.

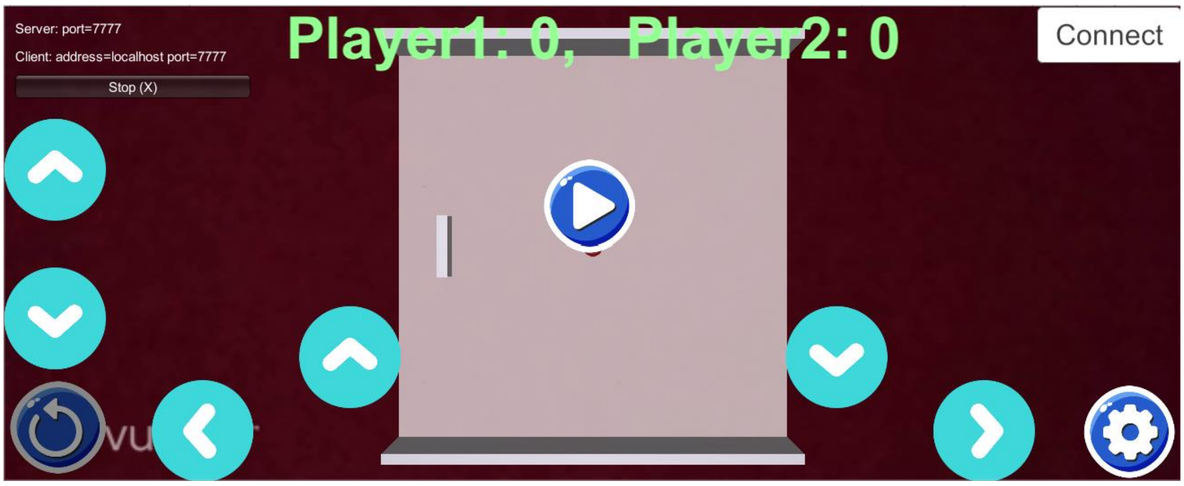


Figure 2: game from top view (before main QR detection)

At this point player1 wall can be seen and moved by the movement button:

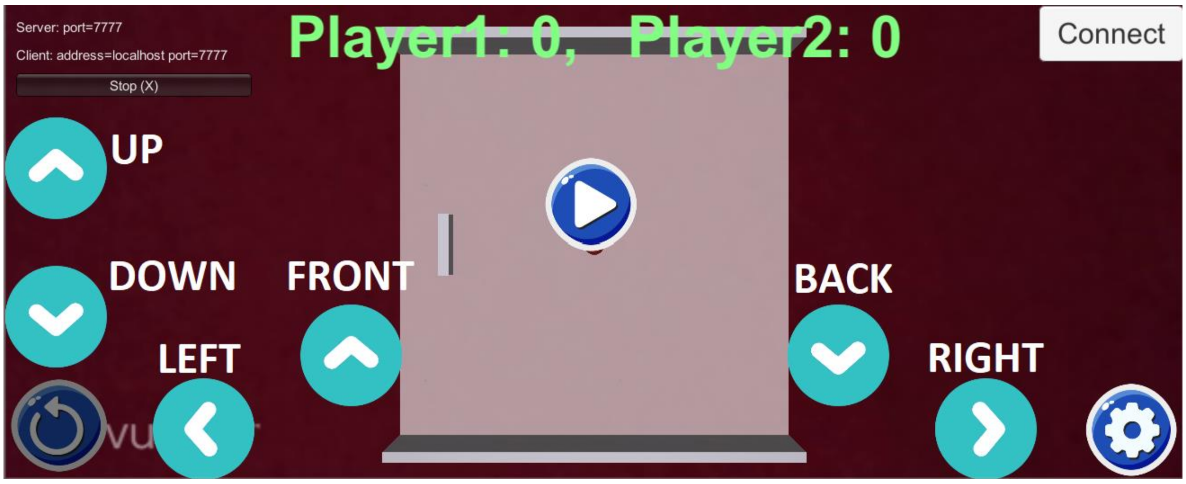


Figure 3: game controls

In the right-bottom corner is a menu button. In the right-top corner is the connect button. In the left-top corner is the networking management menu. All of which will be discussed in detail in different section.

The game can be played with or without drone, the default is to play without a drone, but when the “connect” button is pressed, the device connects to the drone, the drone moves according to the button pressed (up/down/left/right/forward/backward), and the wall is positioned according to the position of the detected drone (the detected QR code on the drone) instead of moving according to the button pressed.

In order to connect to the drone, turn the drone on and wait until the drone green light is blinking, turn on the device’s hotspot, to which the drone will connect using the IP of the device, and press the *Connect* button. The display will change to the following display, in which the connect button is replaced by *Disconnect* button, which disconnect the drone and position the player according to the last known position before the *Connect* button was pressed, and the *TakeOff* button, which starts the drone’s engines and brings the drone to height of about 50cm above the floor. Pressing the *Disconnect* button will also switch back to *Connect* button.

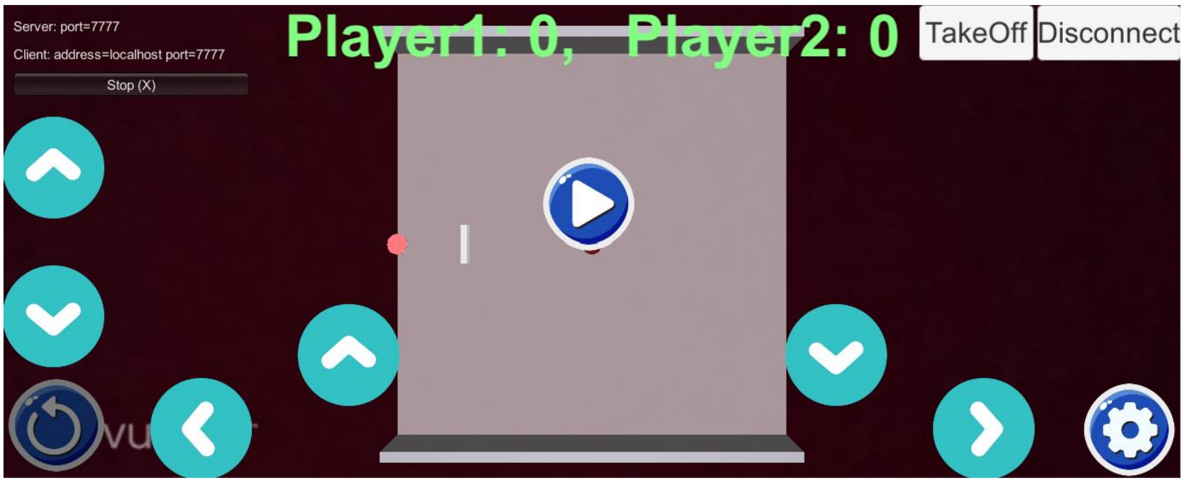


Figure 4: After pressing the Connect button

After the *TakeOff* button is pressed, the *TakeOff* button will be replaced with *Land* button.

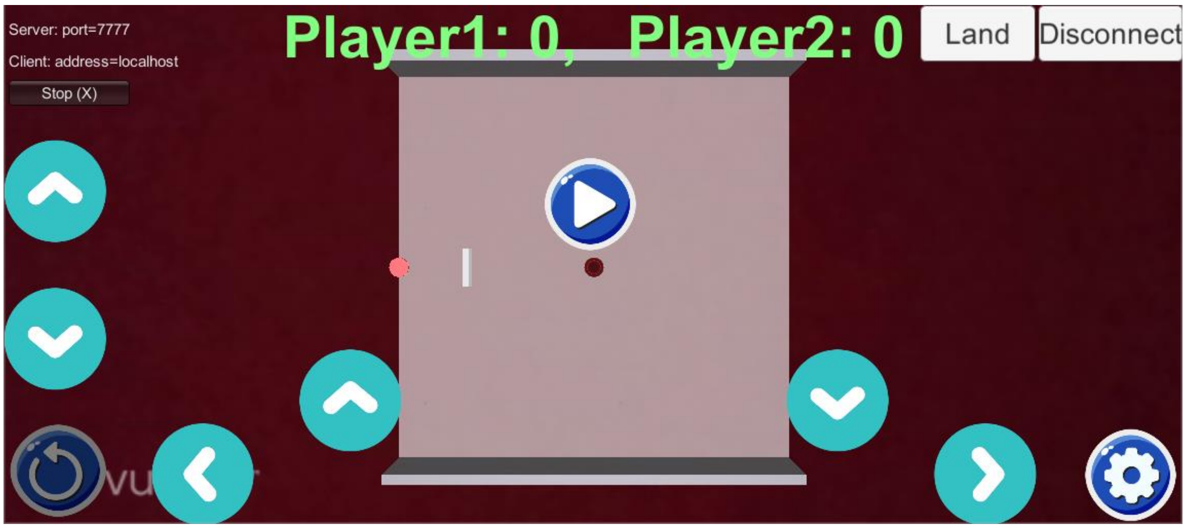


Figure 5: After pressing the TakeOff button

Pressing on the *Land* button will direct the drone to slowly land at its current position, and the button will be switched back to *TakeOff*.

When the Play button is pressed the game starts and the ball moves at a random angle. The ball will change its direction when it hits one of the following: the wall at the top and bottom of the game surface, the two “invisible” walls on the right and left of the surface (in this cases a point will be added to the other player, and the player walls.

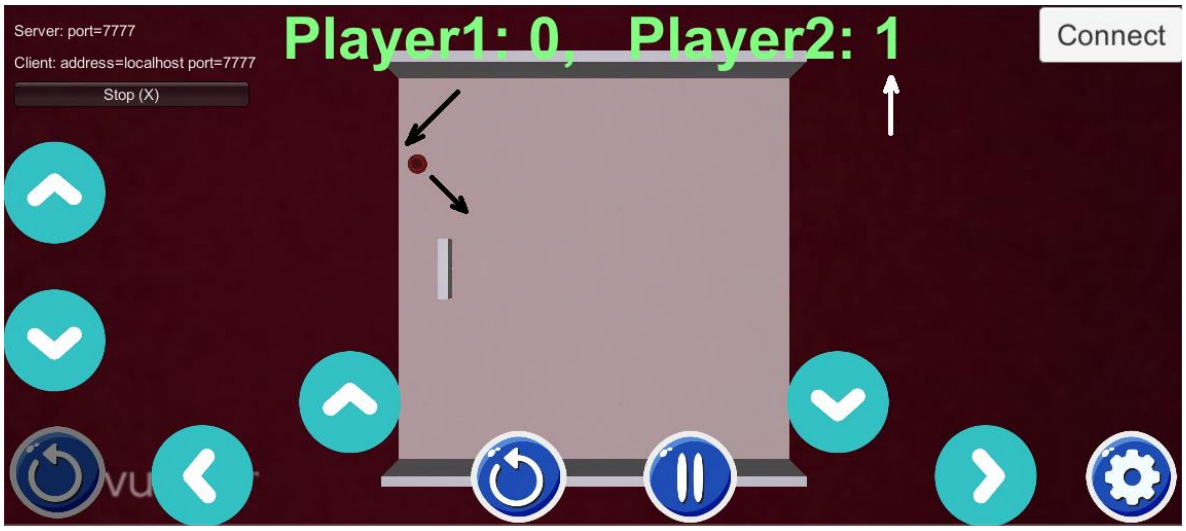


Figure 6: Point made by Player2

The game can be paused by pressing the *Pause* button, or restart to the original ball position, at the center of the surface, by pressing the *Restart* button. Pressing both buttons will pause the game until *Play* button is pressed again.

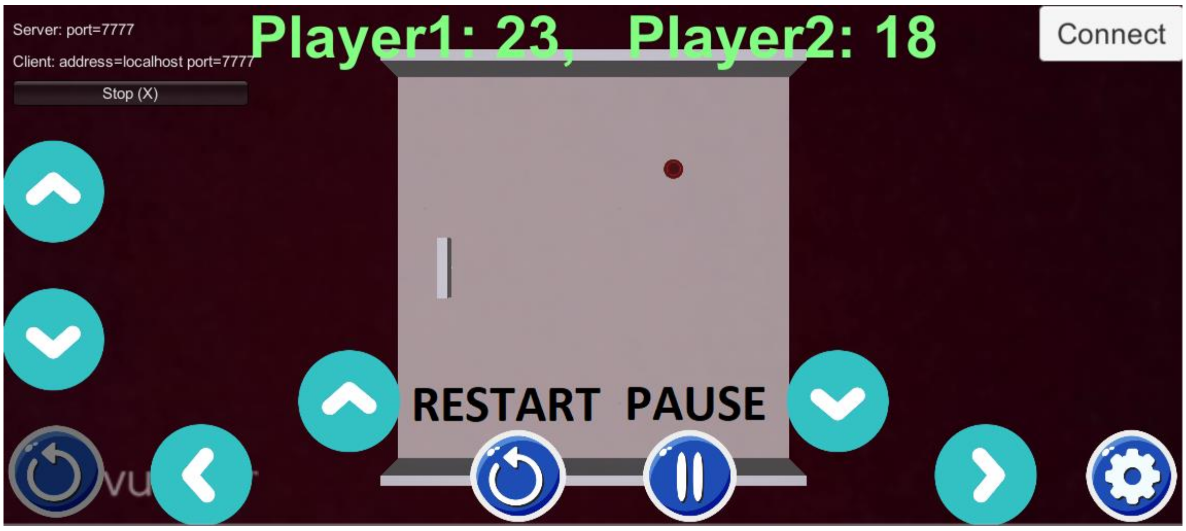


Figure 7: Pause & Restart buttons while in play

Pressing the *Restart* button does not reset the point accumulated by the players. New ball direction is chosen at random.

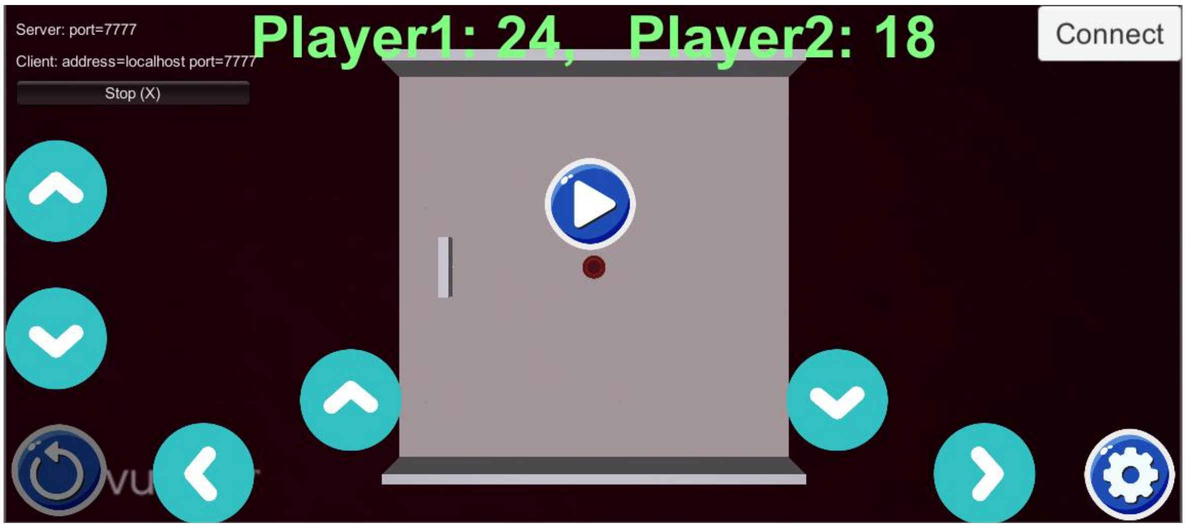


Figure 8: After pressing the Restart button

Since it is sometimes hard to tell from the position of the camera if the player's wall is positioned above, below or in the correct height, there are three markers: red, blue and green. The blue marker indicates that the player is positioned above the position of the ball, right above the marker. The red marker indicates that the player wall is positioned below the surface of the game, right below the marker. The green marker indicates that the player wall and the ball are overlapping.

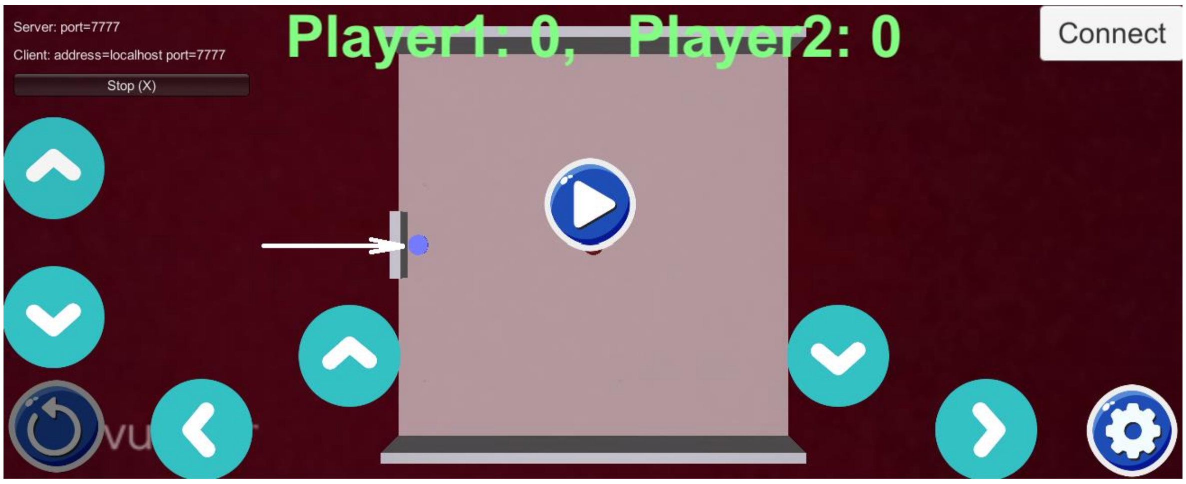


Figure 9: blue marker: player is above the ball

Since the surface is semi-transparent, the player wall can be seen even when it is positioned below the surface. The player wall will appear somewhat dim because of the surface semi-transparency.

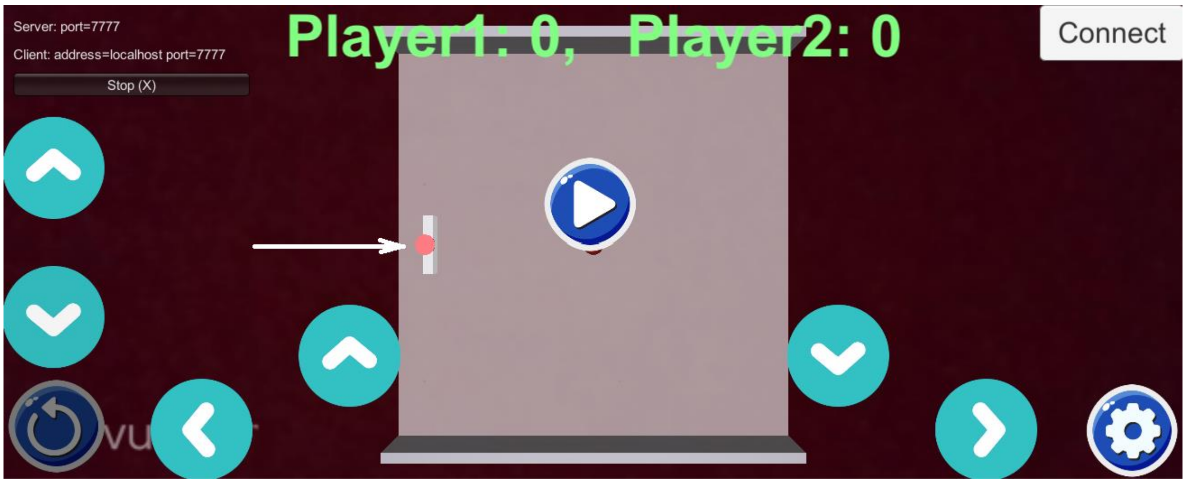


Figure 10: red marker: player is below the game surface

The game is best played in augmented reality mode, as can be seen in the following image, for more detail, refer to section 2.5 below (Virtual & Augmented reality modes).

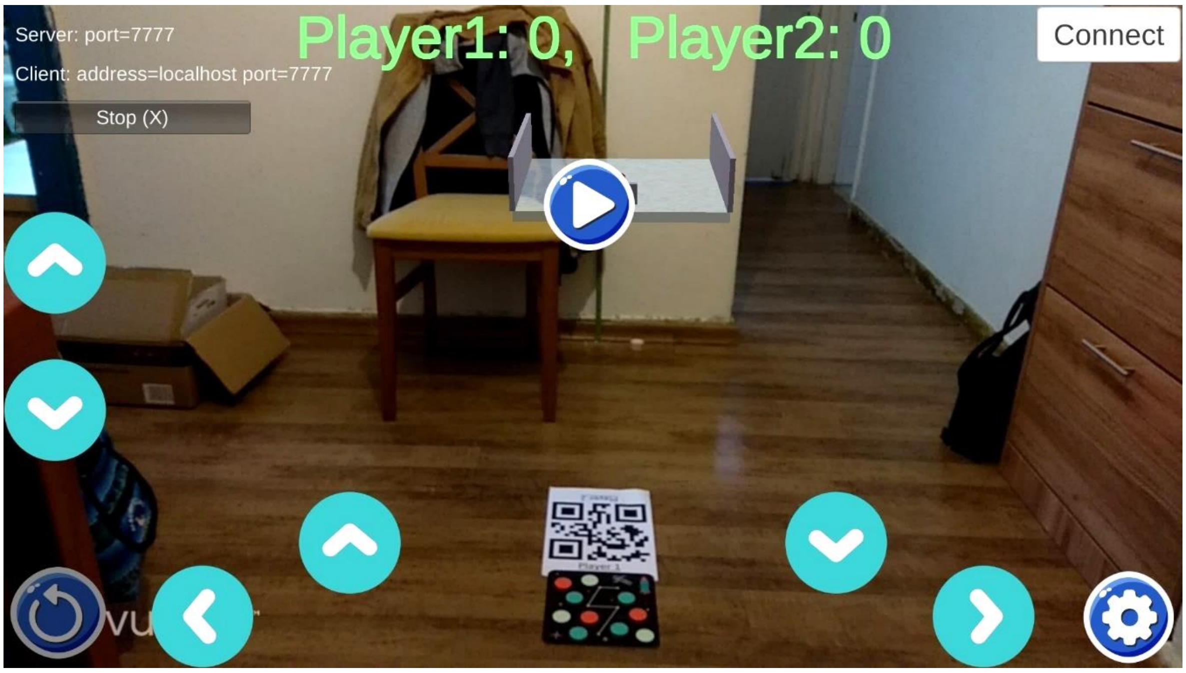


Figure 11: Board is positioned ~60cm above ground in AR game mode

After pressing *Connect* and *TakeOff* the drone is moved to a starting position, and the *Start* button is pressed. To switch back from drone AR game to VR game, press *Land* button and wait for the drone landing before pressing *Disconnect* button.

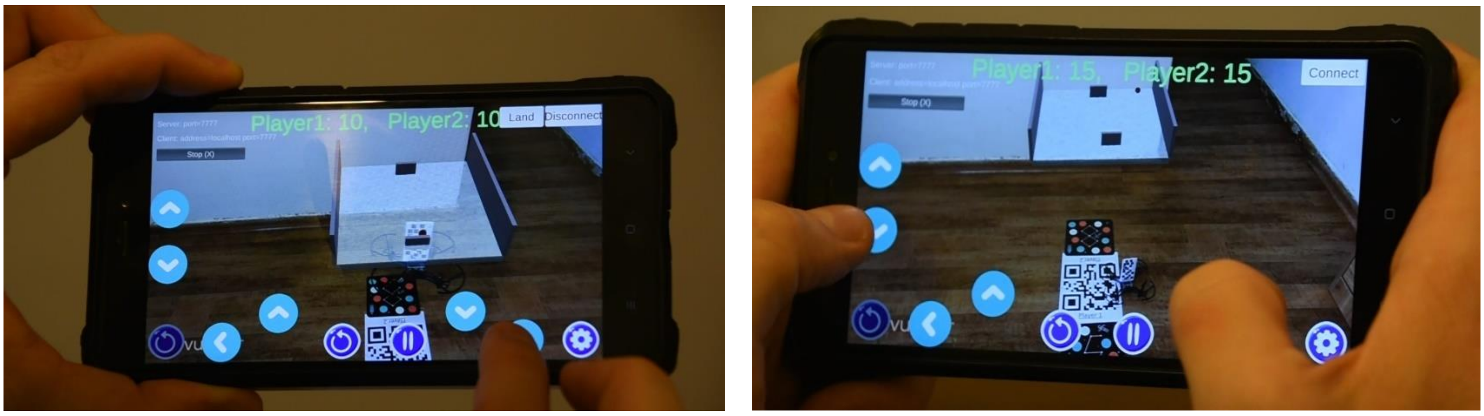


Figure 12: Drone player hit ball (left), After drone landing (right)

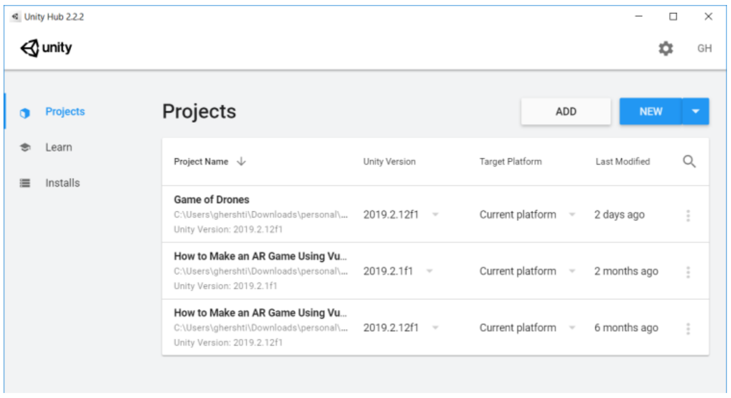
2.2 Unity, Unity Hub, Vuforia, C#, Git

The project work frame was chosen to be Unity 2019.2.12f1 with C sharp (C#), and to keep the project organized, Unity Hub and Git was chosen. Other option that was considered was to use ARKit instead of Vuforia.

Unity was chosen as the main tool of development due to the project goal of playing the game on cell phones devices. Unity is a cross platform development tool, which makes the transition between platforms seamless, so most of the development was done in PC/Standalone mode and finalized on Android. Almost all of the game functionality works on the Android devices (all but the shadow casting, which require higher quality of rendering, which slows down the game).

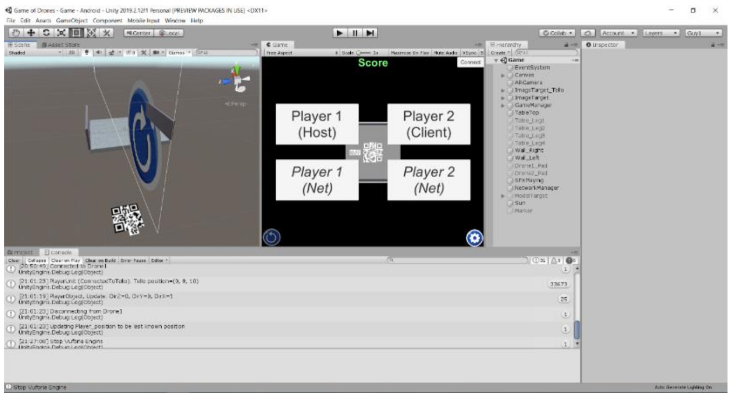
<u>Tools versions used</u>:

Unity	2019.2.12f1
Unity Hub	2.2.2
Vuforia	8.5.9
Visual Studio 2019	16.5.4
JDK	1.8.0_221
NDK	20.0.5594570



Projects

Project Name	Unity Version	Target Platform	Last Modified
Game of Drones C:\Users\ghershti\Downloads\personal\... Unity Version: 2019.2.12f1	2019.2.12f1	Current platform	2 days ago
How to Make an AR Game Using Vu... C:\Users\ghershti\Downloads\personal\... Unity Version: 2019.2.1f1	2019.2.1f1	Current platform	2 months ago
How to Make an AR Game Using Vu... C:\Users\ghershti\Downloads\personal\... Unity Version: 2019.2.12f1	2019.2.12f1	Current platform	6 months ago



```
[20:50:49] Connected to Drone1
UnityEngine.Debug:Log(Object)                                              1
[21:01:23] PlayerUnit (ConnectedToTello): Tello position=(0, 8, 10)
UnityEngine.Debug:Log(Object)                                          33673
[21:01:19] PlayerObject, Update. DirZ=0, DirY=0, DirX=1
UnityEngine.Debug:Log(Object)                                             25
[21:01:23] Disconnecting from Drone1
UnityEngine.Debug:Log(Object)                                              1
[21:01:23] Updating Player_position to be last known position
UnityEngine.Debug:Log(Object)                                              1
[21:27:08] Stop Vuforia Engine
UnityEngine.Debug:Log(Object)                                              1
Stop Vuforia Engine
```

Figure 13: Unity Hub layout (left), Unity layout (right)

To test Vuforia detection and tracking, a simple project was used. This was also a good introduction to using Unity, Vuforia, Git. Vuforia detection and tracking on an 2D A4 target work perfectly, even in less than optimal lighting (low fluorescent lighting, flickering at 50Hz). Vuforia showed no problem in tracking 2 or 3 targets at the same time, with no delays, and even succeeded in detecting and tracking a Parrot Bebop 2 drone using a partial 3D model of the drone (downloaded from the web[29]).
Vuforia is not using colors for detection and tracking and is relying on features that are extracted ahead of time when uploading images using the Target Manager.

Creating GUI in Unity is very simple. It's done from GameObject menu, UI sub menu, selecting button. Basic functionality, such as appear/disappear is done by adding another action to the "On click" and updating GameObject.SetActive (checked checkbox for appearing, and unchecked checkbox for disappear), advanced logic was done in C# functions that can also be called by the "On click".
Objects were created in Unity by adding 3D game objects and assigning materials to them. Game logic and communications were coded in C#.

Some aspects of the game were selected in Unity settings (such as Auto rotation and Allowed orientations for the rotation – only landscape).

2.3 Communication & Networking

There are several ways to establish multiplayer game over a network (wi-fi or over the net). The longest, but most flexible, way is to write a networking driver. The drivers job would be to establish host on one device and client on the other (or client on both devices), and send information from client to host and back, in order to keep both players updated with the latest data (such as ball position, player position, etc.). Another way would to use one of the already existing networking systems written for Unity. Two of these are:

1) Write Server & Client scripts separately, using UnityEngine.Networking library following the following tutorial:
https://www.youtube.com/watch?v=amy3L3pGWH0

2) Using UnityEngine.Networking, NetworkManager and NetworkManagerHUD following these tutorials:
https://www.youtube.com/watch?v=0H_ikQp9aTI
https://www.youtube.com/watch?v=JlKf0h0K5PU

After reviewing all possibilities, the last option was chosen. This way is utilizing Unity network servers (UNet). Some modifications were required in the given code, in order to support different buttons and font size according to screen resolution and adding buttons to allow the user a fast and easy access to wi-fi networking or over the internet. So, the four buttons in the opening screen are actually shortcuts for functionality also given by the NetworkManagerHUD menus.

One concern regarding the UNet is that it is supposed to be deprecated in the coming future (different APIs will stop function 2021-2022, and matchmaker service will stop working in 2022), but migration from old APIs to new APIs is expected to be intuitive[30].

In order to add NetworkManager & NetworkManagerHUD do the following steps:
1. GameObject → Create Empty + change object name to “NetworkManager”
2. Add “Network Manager” component to “NetworkManager”
3. Add “Network Manager HUD” component to “NetworkManager”
4. Add “Network Identity” to Player + check the box “Local Player Authority” (to control player from within, and not from server)
5. Add “Network Transform” to Player (transfer the data of location/rotation over the net)
6. Right click on “Assets” + create folder “Prefabs”
7. Drag & drop the player to “Prefabs” folder
8. Drag & drop the player to “Network Manager” → “Spawn Info” → “Player Prefab”
   - For faster updates: “Player” → “Network Transform” → change “Network Send Rate” to 29 (move slider)

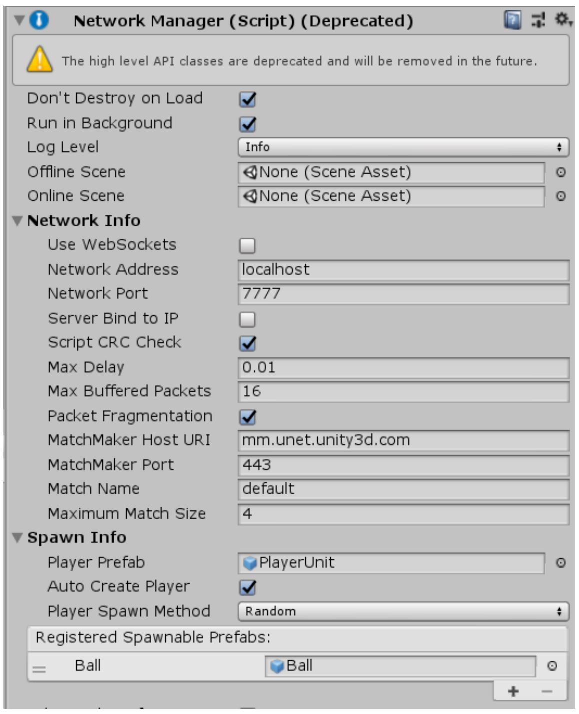
<u>The NetworkManagerHUD menu includes many options. Here I’ll cover the following</u>:

LAN Host	- creates connection as a host server & client (over wi-fi)
LAN Client	- creates connection as a client (over wi-fi)
LAN Server Only	- not used
Enable Match Maker	- open Match Maker menu

<u>Match Maker menu</u>:

Create Internet Match	- creates an Internet match named according to Room Name field
Room Name (field)	- name to be given to match. Default value is “GameOfDrone”
Find Internet Match	- look for an Internet match
Change MM Server	- not used
Disable Match Maker	- return to NetworkManagerHUD main menu

When using Unity networking (UNet), there are basic capabilities that can be used to synchronize the clients and the server. The most common of these is the player prefab. After creating the player, it is moved to *Prefabs* directory, making the player a prefab. At this point the player prefab can be drag-and-dropped to the *Spawn Info* → *Player Prefab*, and checking the *Auto Create Player* checkbox. Once a network game starts, the player will appear on all clients. In order for the player position kept updated on the server and on all other clients, “Network Transform” is added to the player. This script is responsible for keeping all clients updated with the movement of the player. It has several important variables, such as, “Movement Threshold” which defines the minimal movement that will generate an update on the clients. “Transform Sync Mode” which was chosen to be “Sync Transform”. Another important aspect of the script is the number of updates sent in each second. The higher this value is, the faster clients are aware of any change that happened, but more messages are sent, which might slow the game. The value that showed best performance was chosen to be the highest possible value of 29 updates per second.
* *Don't Destroy on Load* should be checked.



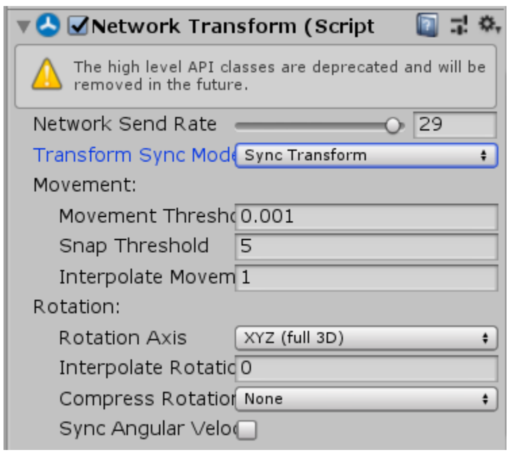
Network Manager (Script) (Deprecated)

The high level API classes are deprecated and will be removed in the future.

Don't Destroy on Load	✓
Run in Background	✓
Log Level	Info
Offline Scene	None (Scene Asset)
Online Scene	None (Scene Asset)
Network Info	
Use WebSockets	☐
Network Address	localhost
Network Port	7777
Server Bind to IP	☐
Script CRC Check	✓
Max Delay	0.01
Max Buffered Packets	16
Packet Fragmentation	✓
MatchMaker Host URI	mm.unet.unity3d.com
MatchMaker Port	443
Match Name	default
Maximum Match Size	4
Spawn Info	
Player Prefab	PlayerUnit
Auto Create Player	✓
Player Spawn Method	Random

Registered Spawnable Prefabs:

Ball	Ball



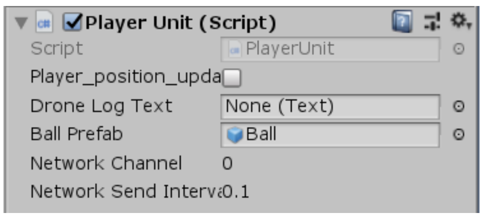
Network Transform (Script

The high level API classes are deprecated and will be removed in the future.

Network Send Rate	29
Transform Sync Mod	Sync Transform
Movement:	
Movement Thresh	0.001
Snap Threshold	5
Interpolate Movem	1
Rotation:	
Rotation Axis	XYZ (full 3D)
Interpolate Rotatic	0
Compress Rotatior	None
Sync Angular Veloc	☐

Figure 14: Network Manager script (left), Network Transform script (right)

The ball position can either be calculated using its' initial position and initial velocity, this way each player calculates the position of the ball independently, instead of constantly sending the ball position. But this proved to work poorly due to minor variations, that accumulated over time to significant change, and led each player to accumulate points differently. Instead, the ball was added as another prefab as part of the main player prefab. Since the ball was not the main prefab, it required to be spawned (see function `CmdSpawnBall` in PlayerUnit.cs).



Player Unit (Script)

Script	PlayerUnit
Player_position_upda	☐
Drone Log Text	None (Text)
Ball Prefab	Ball
Network Channel	0
Network Send Interv	0.1

Figure 15: Ball Prefab as a variable of the Player Prefab

Other options to keep the clients synchronized, are the SyncVars and SyncLists[31]. These are best to keep variables updated on all clients. Every time a client changes one of its SyncVars, it sends a message to the server to update all other clients on this change, and the other clients update their copy of this variable locally. This was tried at first to keep the ball position synchronized on the clients, but it did not work properly, and was not used in the final version of the game.

The score is kept by each client, relying on agreement between clients on the players and ball positions, so no communication is needed in that regard.

2.4 Virtual & Augmented reality modes

The game is best played as Augmented reality game, meaning that the game surface, player walls and ball are rendered on top of the input video feed, coming from the device camera. In order to play in this mode, the device camera should be pointed towards a special QR code around which the entire game is built (see QR code in appendix A). This QR code is used by Vuforia as a "Target".

Adding a target is done in Vuforia Target Manager site[32], by "Add Target" button. When adding a new image, its size must be given as reference for the engine to calculate the position and orientation of the "Target" in the game. Vuforia Target Manager assigns a rating for each given image (0 to 5 stars, in steps of half a star). Higher target rating means faster acquisition, and better tracking. Vuforia engine seems to perform some sort of edge detection, hence, all of the features it uses for aligning the image, are located at corners, as can be seen when pressing the "Show Features" button (the image is slightly scaled, so the features are almost on the corners.

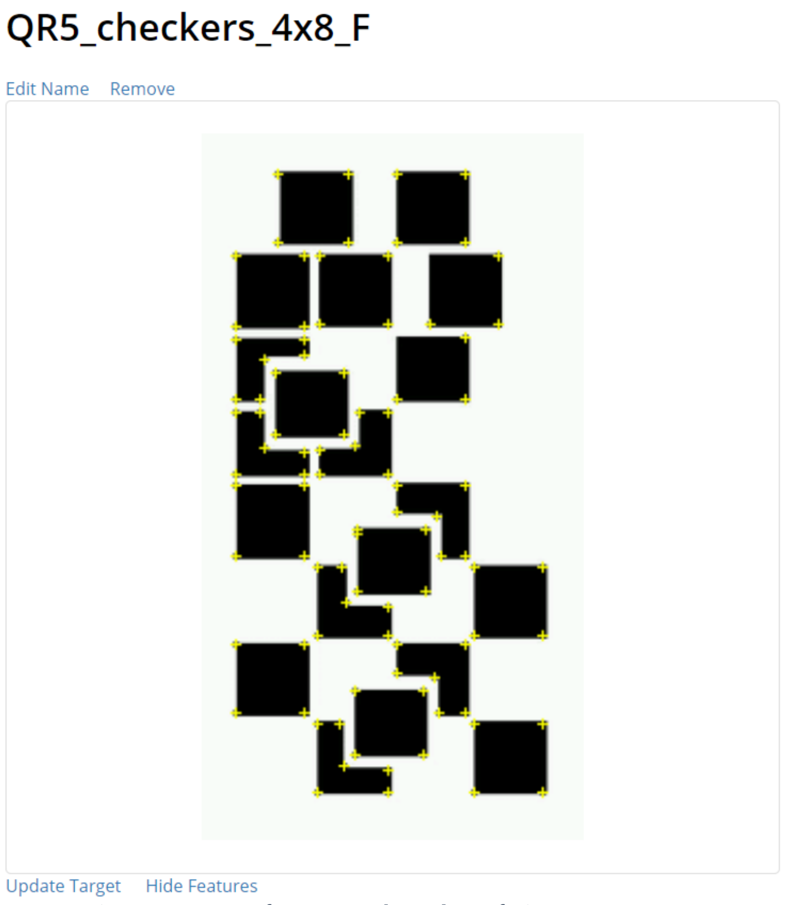


Figure 16: Target features as shown by Vuforia Target Manager

Features are not recognized close to target edges, so a small margin was used when loading the target to the Target Manager. For more detail on how to achieve good targets, refer to Vuforia site[33].

When a target is detected by the device, Vuforia augmented reality engine estimates the position and orientation of the target, compared to the target image in the Unity world, and renders the game accordingly.

In order to switch from VR mode to AR mode, all that is needed is to add the "Board Target" image to the scene. When Vuforia engine detects the target, it will project the entire game world on to the presented live camera feed.

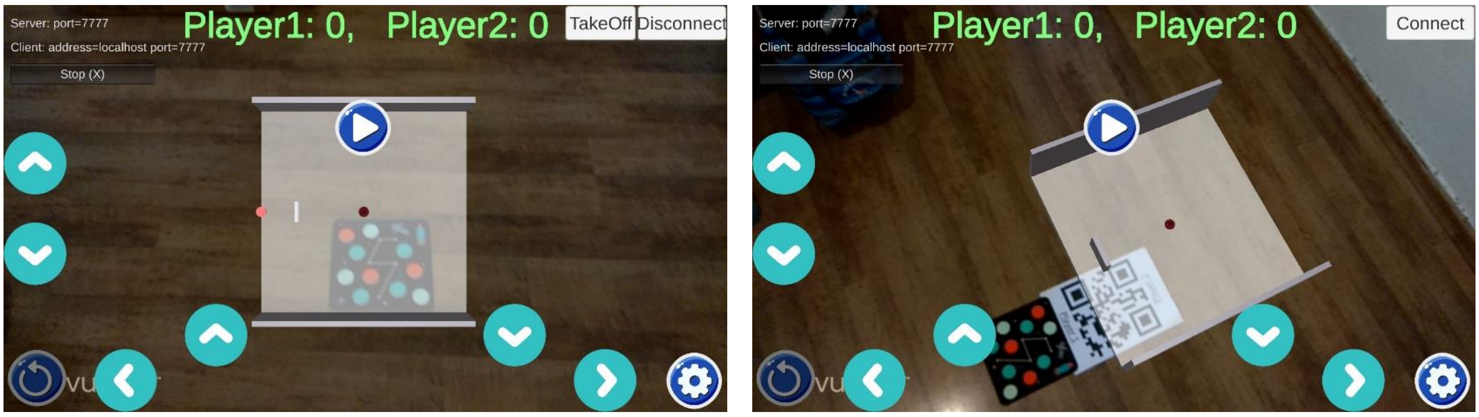


Figure 17: without QR - VR game (left), with QR – AR game (right)

In order to add the player to AR mode, an additional QR is used as the "Player Target", this QR is positioned on the drone using a plastic clip and regular cell-o-type to double sided cell-o-type.
Note that the QR is facing the backend of the drone, since it is pointed towards the device, and it is positioned behind the drone. Moreover, since only the position of the drone is used, and not it's orientation, it is possible to print the QR on both sides, so the player position would not be limited to be behind the drone. In this case, going from the back facing QR to the front facing QR will cause Vuforia engine to lose track of the QR, until it is found again.

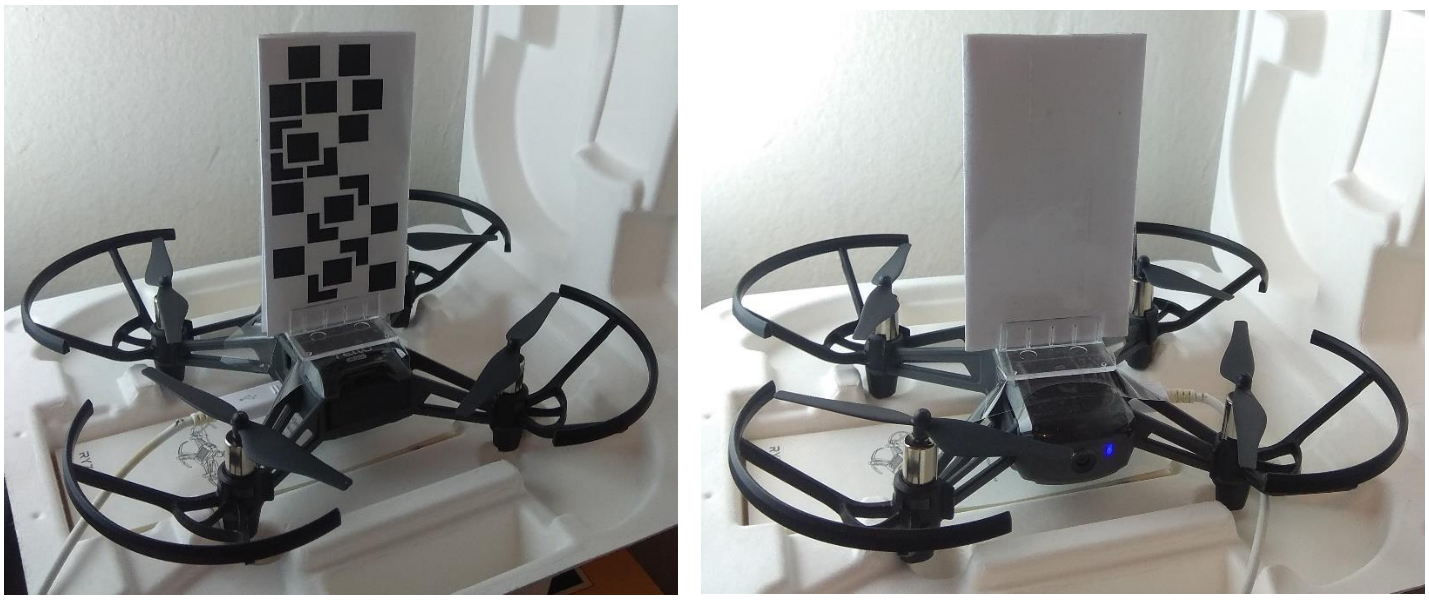


Figure 18: Drone with QR – from back (left), Drone with QR – from front (right)

It’s best to position the QR as shown in the following image, this was found to be the center of mass of the drone. The QR & plastic clip weight is about TBD, and hardly change the drone balance, movement or time in the air.

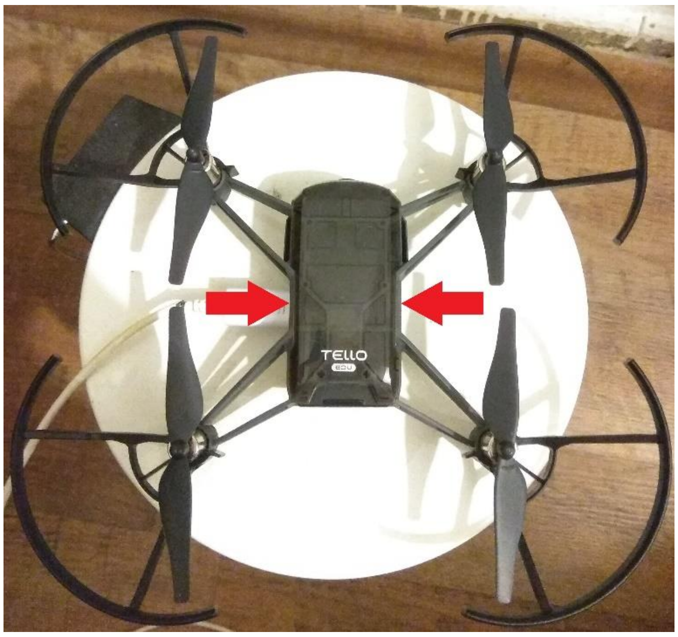


Figure 19: Drone center of mass - player QR position

When the device is not connected to the drone, the player wall is positioned relative to the board in a default location. Once the *Connect* button is pressed, and the player target is found, the player wall is positioned at the target position.

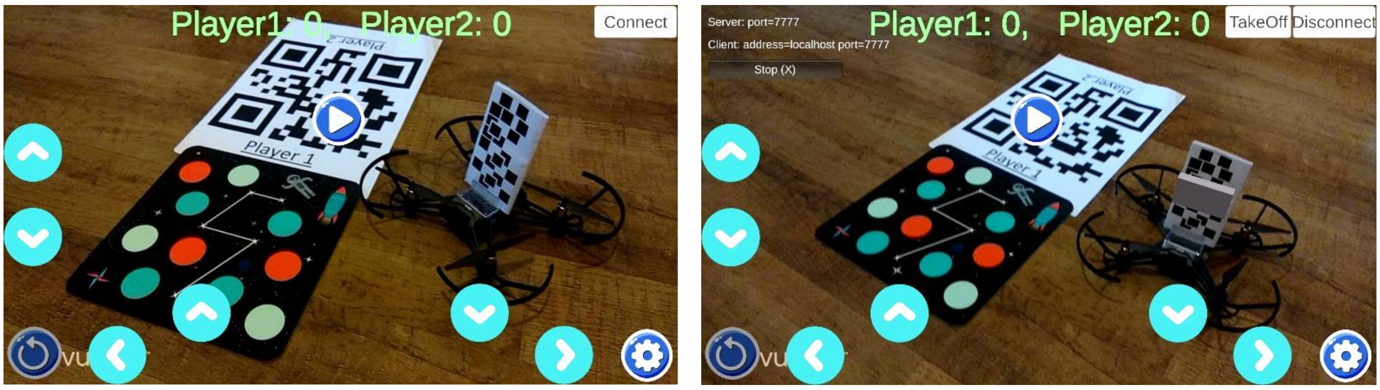


Figure 20: Drone not connected (left), Drone connected and drone QR found (right)

2.5 "Pong" game logic

The game logic consists of the ball movement and the scoring. The ball initial 3D vector of movement (called *direction*) is selected at random and normalized. In order to create valid directions, first the sign of the x-axis is chosen with 50% probability to be either side, so the ball can start going in both players direction, then the initial x-axis and z-axis values are randomized between 3-5, and the x-axis value sign is used. The y-axis is set to be 0, so the ball will not move in y-axis, since it will make the game much more complex to play, but this can be easily changed here. The direction 3D vector is normalized, and the ball speed vector is the multiplication of *direction* vector and *speed* scalar (speed was chosen to be 0.7 to give smooth ball movement). The ball initial position is at the center of the board in x/z-axis, and at ball radius height above the board to appear hovering above the board. The ball movement is done by applying Translation transformation on the ball in the following manner:

transform.Translate(direction * speed * Time.deltaTime);

The Time.deltaTime ensures that if more calculations were done between ball updates, and the updates are not done in constant time intervals, the ball movement will take the time that passed into consideration and the ball will appear to move in a constant speed.

When the ball reaches one of the board sides, defined as positions `Lo_Bound_X, Lo_Bound_Y, Lo_Bound_Z, Hi_Bound_X, Hi_Bound_Y, Hi_Bound_Z`, the ball direction x/y/z will be negated accordingly.

When the ball collides with one of the player prefabs, `OnTriggerEnter` function is called. In this function, the collision point is used, with the ball position, and the ball radius, to calculate the new ball direction.

If the *Restart* button is pressed, the ball will re-appear in the initial position, and a new ball direction will be randomized. If the *Pause* button is pressed, the ball will stop moving (only the Host can press *Restart* and *Pause*).

When the ball reach either side of the board in y-axis (`Lo_Bound_Y`, `Hi_Bound_Y`), Player2 or Player1 will receive a point, respectively.

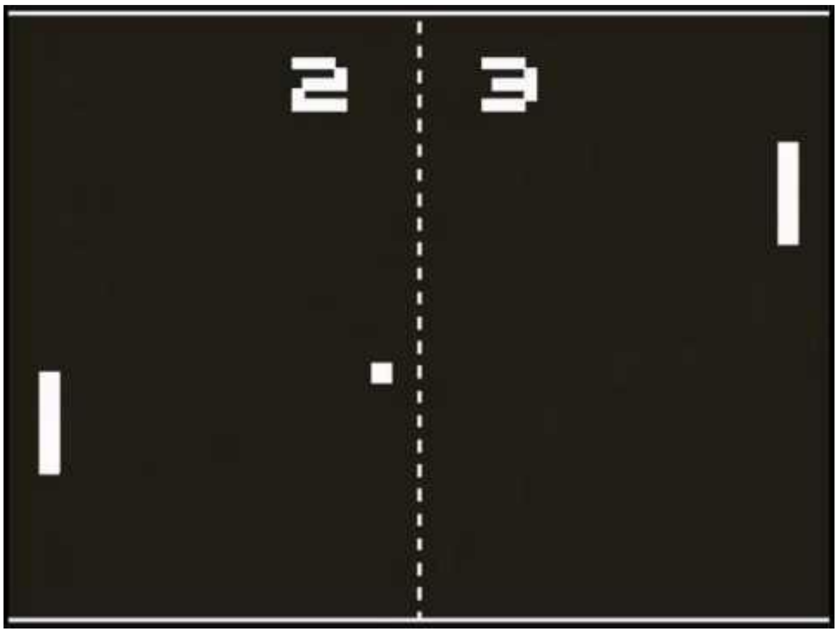
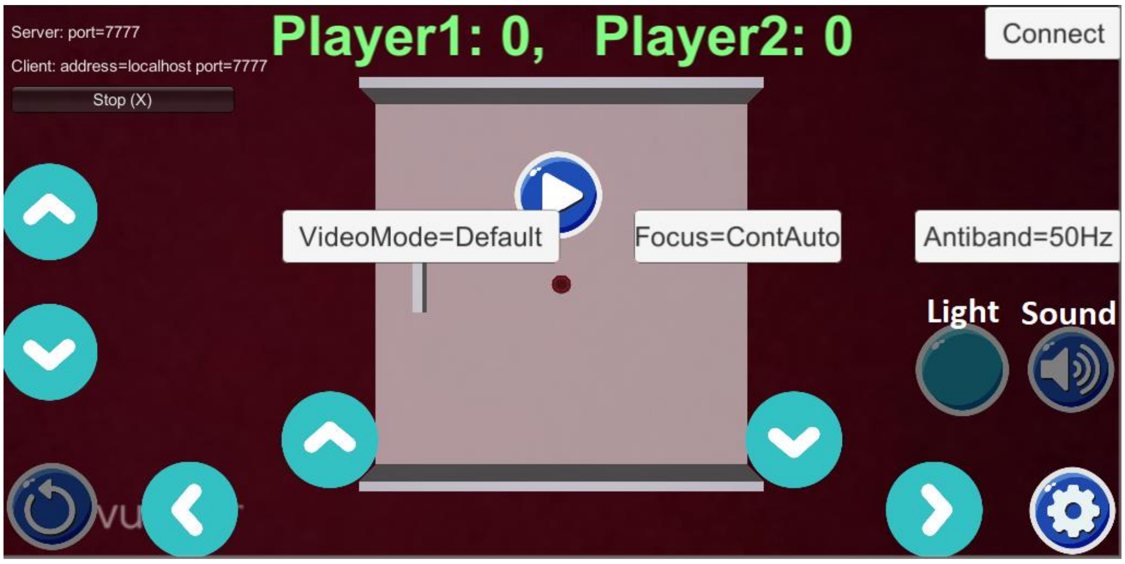


Figure 21: Pong video game machine (left), Pong screen shot (right)

2.6 Extras (Light, Sound, Focus, Video mode, Antiband, Rotate)

Pressing the cog-wheel icon in the right-bottom corner will open the Menu


Server: port=7777

Client: address=localhost port=7777

Player1: 0, Player2: 0

VideoMode=Default

Focus=ContAuto

Antiband=50Hz

Figure 22: Menu

The menu has 5 options (The first option was chosen as the default value):

1. VideoMode[34]:

• Default	- A compromise between Speed and Quality
• Speed	- Optimize for speed. Quality of the video background could suffer
• Quality	- Optimize for quality. Application performance could go down

2. Focus[34]:

• ContAuto	- Continuous autofocus mode
• Infinity	- Focus set to infinity
• Macro	- Macro mode for close-up focus
• Normal	- Default focus mode
• TrigAuto	- Triggers a single autofocus operation

3. Antiband[35]:

• 50Hz	- Anti-banding enabled for 50Hz flickering
• 60Hz	- Anti-banding enabled for 60Hz flickering
• Off	- No Anti-banding

4. Light[35]:

• Off	- Flash light is off
• On	- Flash light is on

5. Sound[35]:

• Off	- Sound F/X are off
• On	- Sound F/X are on

ISO and camera exposure were to be chosen by the device automatically, for more detail on camera possible configurations, refer to Vuforia Advanced Camera API[35].

In addition to drone movement in 3 axes, the drone can also turn left or right. Although this functionality is not required by the game, it was coded for future use. To enable drone turning mode, press the button on the bottom-left corner.

2.7 Drone communication & control

The selected drone was DJI Tello Edu[36] for its simplicity, and small size, which makes it convenient for indoor entertainment. The drone can be controlled via wi-fi connection in two options: as a wi-fi hotspot, which means that the mobile device needs to connect to it, or by turning the mobile device to a hotspot, and the drone connects to it. The second option was chosen, to allow connecting the mobile device to the internet, hence enabling the internet version of the game, which proven to be a key functionality in times of the COVID-19 quarantine.

Connection to drone is done via its IP address (configured in ConnectButton_handler.cs). To find the Tello IP address, open a dos shell window and type the following command:

```
    for /L %i in (2,1,254) do ping 192.168.43.%i -n 1 -w 2
```

This command will ping all IPs 192.168.43.x, and the only response will come from the Tello.

The communication to the drone is done by sending text messages with the commands using UDP protocol, for example:

`"takeoff"`	- Start engines, takeoff and hover at ~60cm above floor
`"land"`	- Land drone and stop engines

Moving the drone is done by setting 1 to `GameManager._telloClient.TurnLeft`, to turn left, and in a similar way all of the following directions:
`TurnRight, MoveLeft, MoveRight, MoveDown, MoveUp, MoveBackward, MoveForward`, which set Roll, Pitch, Yaw & Throttle values of [-100,100]. For simplicity of the game, the drone speed was set on a constant value, set in MoveSpeeFactor in the GameManager.cs.

The drone operates better when flying above one of the landing pads (“Mission Pads”), which are provided with the drone. The drone can identify the different pads, using its down facing camera, and execute a command according to the number written on them. This functionality is not required in the game, but since the drone can identify the pad, it should improve drifting issues. This functionality is possible with Tello Edu that are using the Tello SDK 2.0, for more detail, refer to Tello Edu User Guide[37].

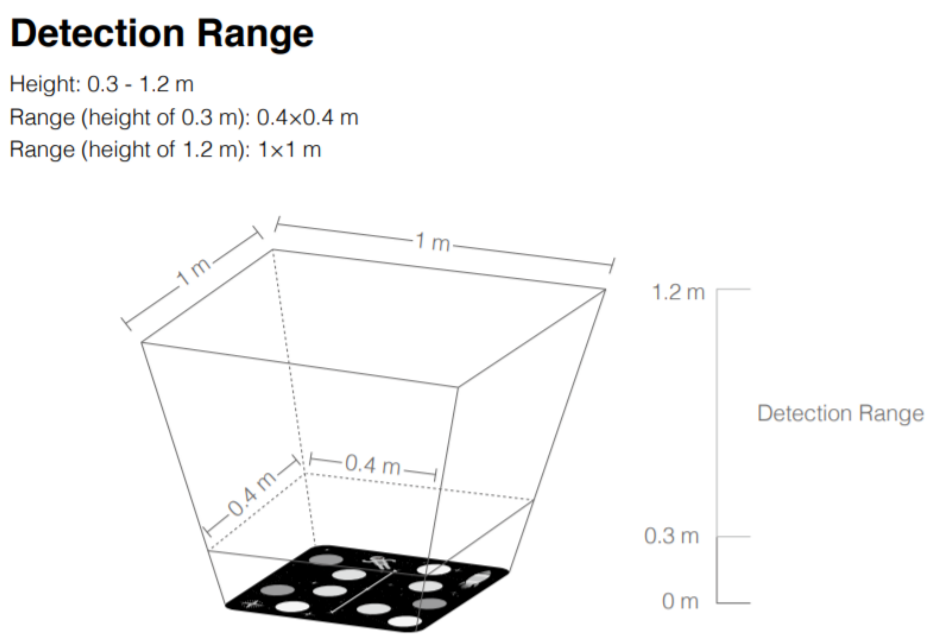


Figure 23: “Mission Pad” Detection Range (taken from Tello User Guide)

3 Results

3.1 Game and devices communication

Communication between two cellular devices or cellular device and computer over the internet was somewhat slower than when using wi-fi, but it did not make the game harder to play. Sometimes small “jumps” of the player wall or the ball were seen on the client device. The host device didn’t have any of these issues.

NetworkManagerHUD actions, such as establish a game over the internet, was done instantaneously. Other than the fact the button size and font size did not scale to fit different resolutions, which was easily fixed in NetwrokManagerHUDNew, it performed perfectly.

Playing the game in 2D is very challenging, since it’s not always easy to understand the position of the ball in reference to the player wall (It’s hard to tell the depth from the change in size). The blue/red/green marker made it easy.

The score was kept the same on both devices (two cellular devices or a cellular device and a computer), even if the player had moved right before the ball hit it.

Playing with sound effect was more fun, but became overbearing after a short while, requiring the addition of sound on/off button in the menu.

Pause and *Restart* buttons worked fine, but sometimes after *Restart* was pressed the ball seemed to change its original height, this was fixed when starting the app again.

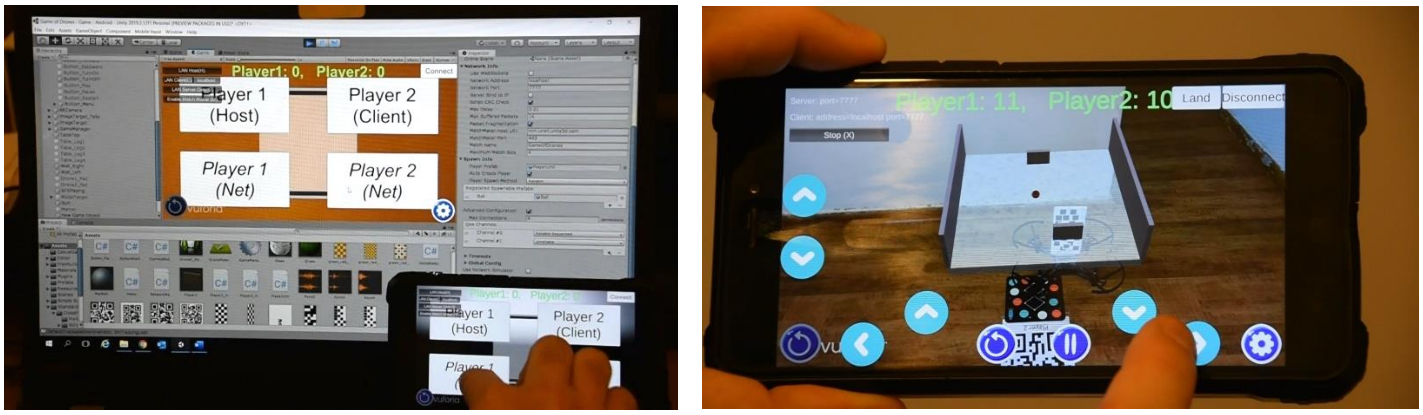


Figure 24: Establish computer-cellular game (left), controlling the drone (right)

3.2 Drone control and drift

Connecting and disconnecting from the drone worked fine and very fast. The *TakeOff* and *Land* commands worked perfectly.

The drone reacted very fast to the movement commands, requiring very short button presses. Longer presses caused the drone to move too fast (using the speed factor of 0.25). Going down was slightly faster as opposed to moving up, while left, right,forward and backward movements had the same speed (this was only observed in long presses).

The drone did not stay in a constant location when no movement command was issued. It had constant small perturbations in position (±3 cm), which made the playing much more difficult. Placing the “Mission Pad” underneath it did not solve the issue. Closing all windows/fans helped a little, but even when the drone did not experience external force, it did not keep its exact position.

The drone had a drift even when flying above the “Mission Pad” and with no external force applied to it. Adjusting the drone QR improved the drift a lot, but it still had a small drift.

Rarely after disconnecting from the drone, it could not connect again, and required reboot of the cellular device and shutting the drone down before it worked again.

Drift measurements were done by pressing the *TakeOff* button, waiting for 45sec and pressing *Land* button, without pressing any other movement buttons. In theory the drone should land at the same position from which it took off. The measurement was done with the same conditions (lighting, external forces, etc.)
Different markers were tried (+ shape instead of – shape), but the drift results were similar.

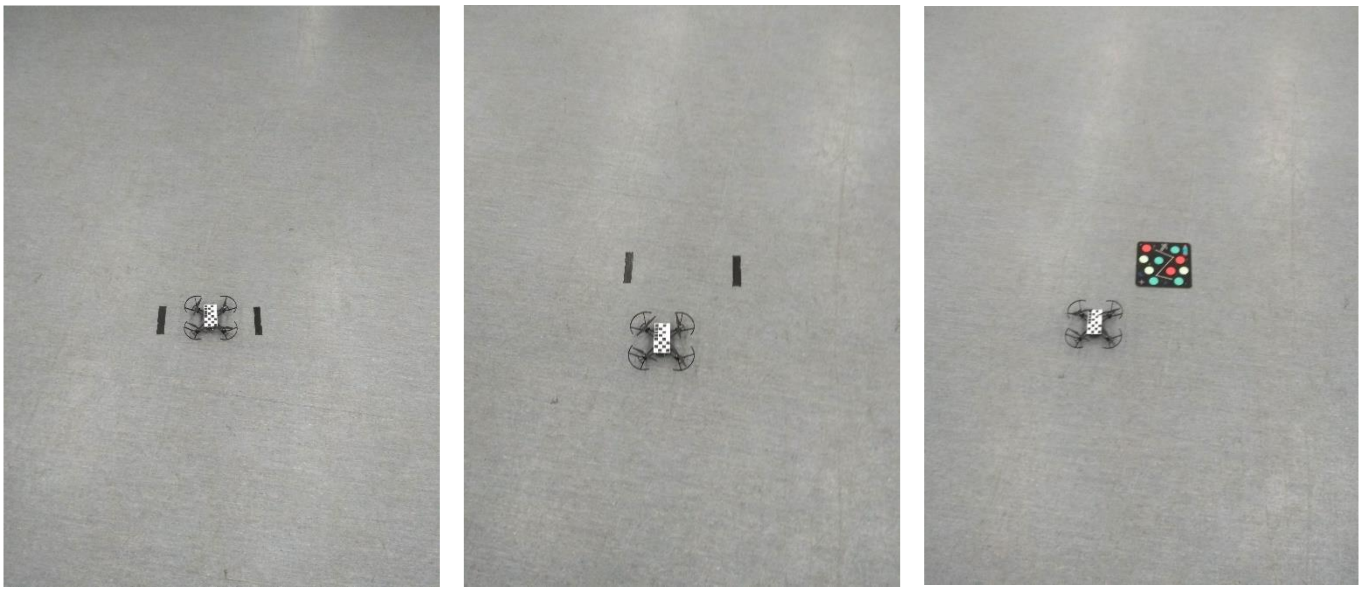


Figure 25: Drift measurement init (left), land position1 (middle), land position2 (right)

3.3 Vuforia target acquisition & tracking

When playing in low fluorescent light the target that graded below 3-stars did not perform well, while 5-star targets worked well, although it took a longer time to initially acquire the target. Once the target was acquired, the tracking worked well both for 3-stars and 5-stars targets. Using anti-banding 50Hz slightly improved the speed of acquisition and the quality of the tracking, other options of anti-banding did not help.

Using the device flash to create more light did not improve the acquisition and tracking.

Vuforia tended to crush after changes made during run, but after last changes were made it did not crush while playing the game from the cellular device or the computer.

Small movements of the cellular device caused flickering in the tracking. The tracking was correct, but it made playing much harder.

Two board QR codes were tried as targets. Both had the maximal 5-star rating. 10 different drone targets were tried, and received different ranking from 0-star to 5-star. These targets were tested for acquisition distance & speed and tracking capabilities.

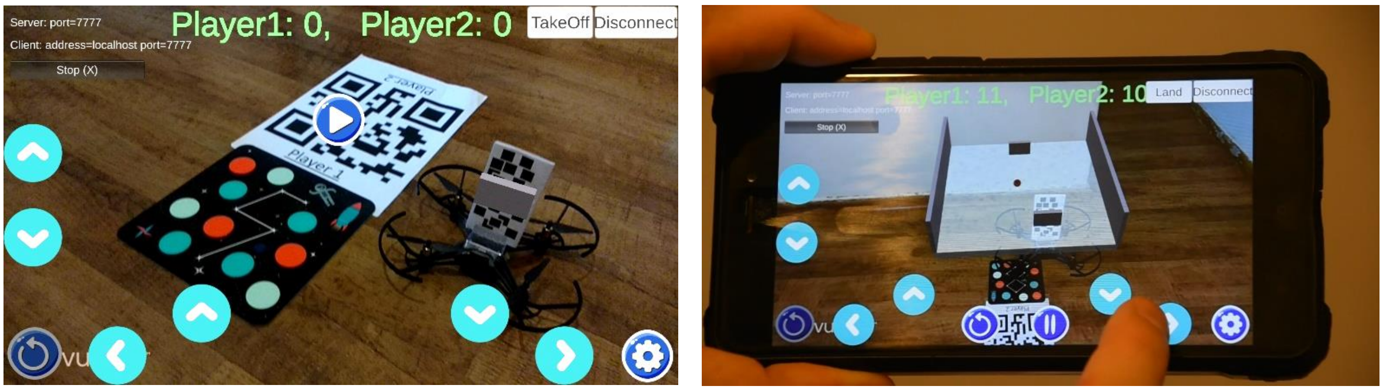
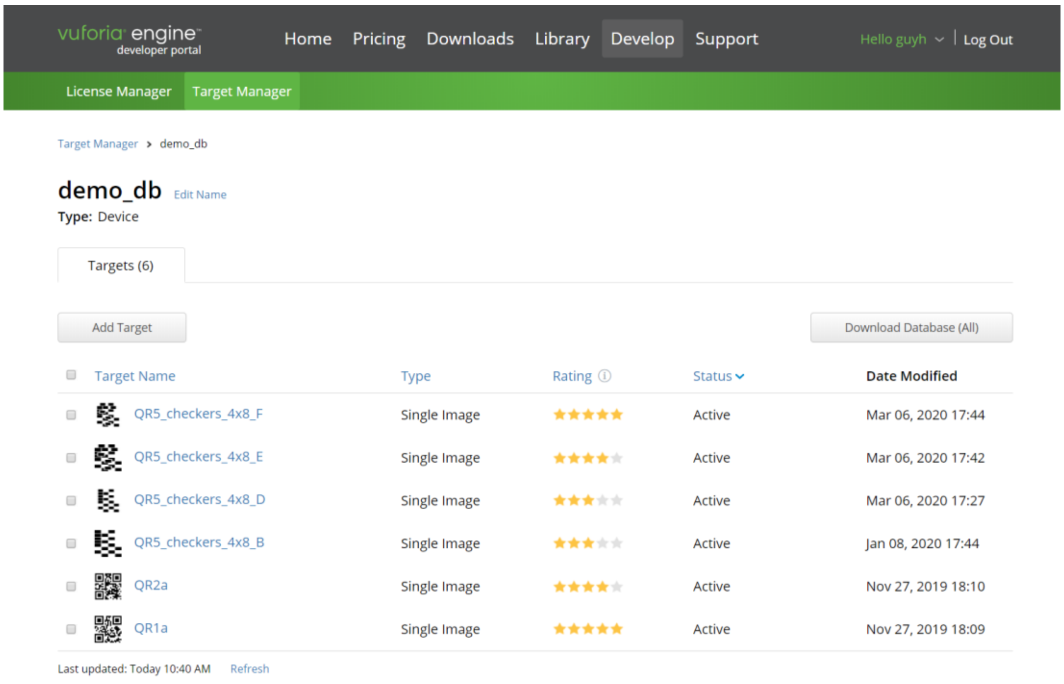


Figure 26: Vuforia drone target tracking (left), drone & board target tracking (right)



vuforia engine developer portal

Home | Pricing | Downloads | Library | Develop | Support | Hello guyh | Log Out

License Manager | Target Manager

Target Manager › demo_db

demo_db Edit Name

Type: Device

Targets (6)

Add Target | Download Database (All)

Target Name	Type	Rating	Status	Date Modified
QR5_checkers_4x8_F	Single Image	★★★★★	Active	Mar 06, 2020 17:44
QR5_checkers_4x8_E	Single Image	★★★★☆	Active	Mar 06, 2020 17:42
QR5_checkers_4x8_D	Single Image	★★★☆☆	Active	Mar 06, 2020 17:27
QR5_checkers_4x8_B	Single Image	★★★☆☆	Active	Jan 08, 2020 17:44
QR2a	Single Image	★★★★☆	Active	Nov 27, 2019 18:10
QR1a	Single Image	★★★★★	Active	Nov 27, 2019 18:09

Last updated: Today 10:40 AM Refresh

Figure 27: Vuforia target manager – targets ratings

4 Discussion and Conclusions

Both wi-fi and internet game work in a satisfactory speed with minor lags. There has been a problem to establish an internet game on the “free-tau” network, probably due to a TAU firewall issue, but was running smoothly on a cellular hotspot, that is mandatory for drone communication anyway.

The DJI Tello Edu drone showed some position perturbations that made the game a little hard, and less enjoyable. This could be easily improved by increasing the player wall.

Drone control was good enough to allow enjoyable experience, but the drone drift required constant user corrections. This can be improved by a close loop control on the drone position, as acquired by Vuforia, and the drone control inputs from device, meaning that while the *Left* button is not pushed, for example, any drift to the left should be canceled. Outdoor game was not tested, but considering the light weight of the drone, it’s expected to suffer major drifts, caused by the wind. Making this drone a poor candidate for outdoor gaming.

The drone battery was sufficient for about 3-5 minutes of game before it required 1.5 hours of charging, making this drone less than ideal for gaming.

Small movements of the cellular device caused image jitter. This can probably be improved by image stabilization done by software, or by using a cellular device with a better camera (Xiaomi Redmi 4[38] was used, which is a cheap and old device). This device was chosen in order to guarantee that the game does not computationally heavy and can run on practically any cellular device.

Vuforia target acquisition depends heavily on lighting conditions and the target quality. While the lighting might not be controlled, the target pattern can be further improved. Having more feature point improves the acquisition time and the quality of the tracking. Using the device light slightly improved the target acquisition, but it seems that the light flickering at 50Hz had a stronger effect. This was slightly improved by the anti-banding, but even with it there was evident flickering which degraded the acquisition.
Low rated targets had a lower maximal acquisition distance, compared to high rated targets. These targets also showed more degradation when tested in poor lighting, compared to high rated targets.

The board target had to be in view at all times, to allow correct game rendering. This can be avoided by using SLAM techniques for localization. These techniques require to make power draining calculations on the drone processor, and send video stream, that is also power costly, which will reduce the Tello drone fly time significantly. Other drones will be more suitable for this task.

Continuous autofocus mode gave the best performance, and the default video was the only video mode that worked on the Xiaomi Redmi 4, although other video modes should have been working better. This could be a limitation of this specific cellular device.

Figuring out the exact position of the drone with respect to the board proved difficult. This was solved by adding the blue/red/green markers.

Using Unity & Vuforia proved to be easier than expected. There is a lot of online material regarding working with Unity, multi-player networking, creating GUI, and using Vuforia, making Unity & Vuforia ideal for this sort of project.

Future work

After meeting all of the project goals of creating an AR game using drones, and understanding the limitations of the project, there some additions that can improve the project in this area. Other improvements are nice to have, but can also be considered.

The following work is suggested as continuation projects:

- Replace board target with SLAM
- Change drone acquisition from QR code to drone 3D model -
  (a proof of concept of was performed using un-accurate 3D model, which gave longer acquisition time. Tracking was just as good as with QR code)
- Reduce drone drift, using acquired drone position and the commands given to it
- Limit drone movement to be always in a certain area
- Create different types of games

5 Bibliography

[1] Van Krevelen, D.W.F., & Poelman, R. (2010). A survey of augmented reality technologies, applications and limitations. The International Journal of Virtual Reality, 9(2), 1–20

[2] Khan W.A., Abdul Raouf A., Cheng K., (2011). Virtual Manufacturing. London (UK): Springer-Verlag

[3] Milgram P., Kishino A.F., (1994). Taxonomy of Mixed Reality Visual Displays. IEICE Transactions on Information and Systems, E77-D(12), 1321–1329

[4] Tamura H. (2002). Steady steps and giant leap toward practical mixed reality systems and applications. Proceedings of the International conference on Virtual and Augmented Reality (VAR 2002), 3–12.

[5] Johnson L., Smith R., Willis H., Levine A., Haywood K., (2011). The 2011 Horizon Report. Austin, Texas: The New Media Consortium. Available online at http://www.nmc.org/pdf/2011-HorizonReport.pdf

[6] Höllerer, T.H., & Feiner, S.K. (2004). Mobile augmented reality. In: Karimi, H. & Hammad, A. (eds.), Telegeoinformatics: Location-based computing and services, 21, 1–39. London (UK): Taylor and Francis Ltd

[7] Van Krevelen, D.W.F., & Poelman, R. (2010). A survey of augmented reality technologies, applications and limitations. The International Journal of Virtual Reality, 9(2), 1–20

[8] Henderson S.J., Feiner, S. (2009). Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2009), Orlando (USA), 19-22 October 2009, pp.135-144

[9] Regenbrecht H., Baratoff G., Wilke W. (2005). Augmented reality projects in the automotive and aerospace industries. IEEE Computer Graphics and Applications, 25(6), 48–56

[10] Vogt S., Khamene A., Sauer F. (2006). Reality augmentation for medical procedures: System architecture, single camera marker tracking, and system evaluation. International Journal of Computer Vision, 70(2), 179–190

[11] STARNER, T., LEIBE, B., SINGLETARY, B., AND PAIR, J. 2000. MIND-WARPING: Towards creating a compelling collaborative augmented reality game. In *Proceedings of the 5th International Conference on Intelligent User Interfaces*, ACM, New York, 256–259

[12] ODA, O., LISTER, L. J., WHITE, S., AND FEINER, S. 2007. Developing an augmented reality racing game. In Proceedings of the 2nd international Conference on Intelligent Technologies for Interactive Entertainment, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)

[13] CHEOK, A. D., GOH, K. H., LIU, W., FARBIZ, F., FONG, S. W., TEO, S. L., LI, Y., AND YANG, X. 2004. Human Pacman: A mobile, wide-area entertainment system based on physical, social, and ubiquitous computing. Personal Ubiquitous Comput. 8, 71–81

[14] MATYSCZOK, C., RADKOWSKI, R., AND BERSSENBRUEGGE, J. 2004. AR-bowling: Immersive and realistic game play in real environments using augmented reality. In Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ACM, New York, 269–276

[15] OHSHIMA, T., SATO, K., YAMAMOTO, H., AND TAMURA, H. 1999. RVBorder Guards: A multi-player mixed reality entertainment. Trans. Virtual Reality Society of Japan 4, 699–705

[16] OHSHIMA, T., SATOH, K., YAMAMOTO, H., AND TAMURA, H. 1998. AR2 Hockey: A case study of collaborative augmented reality. In Proceedings of the IEEE 1998 Virtual Reality Annual International Symposium, IEEE, Washington, D.C., 268–275

[17] Eisenbeiss, H. (2004). A mini unmanned aerial vehicle (UAV): System overview and image acquisition. International Archives of Photogrammetry. Remote Sensing and Spatial Information Sciences, 36(5/w1)

[18] SPARKED: A Live Interaction Between Humans and Quadcopters. (2014). www.youtube.com/watch?v=6C8OJsHfmpI

[19] Takahashi, D. (2016). Air Hogs Connect: Mission Drone debuts with augmented reality game. VentureBeat. http://venturebeat.com/2016/11/01/air-hogs-connect-missiondrone-debuts-with-augmented-reality-game/

[20] AR.Race: 1,2,3… GO! (2012). www.youtube.com/watch?v=Faww1TMs2n8

[21] AR. Drone localization with visual markers. (2013). https://vimeo.com/66909596

[22] Whyte H D. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. Robotics and Automation Magazine, 2006, 1–9

[23] Bailey T, Durrant-Whyte H. Simultaneous localization and mapping (SLAM): part II. IEEE Robotics & Automation Magazine, 2006, 13(3): 108–117

[24] Davison A J, Reid I D, Molton N D, Stasse O. MonoSLAM: real-time single camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29(6): 1052–1067 DOI:10.1109/tpami.2007.1049

[25] Zou D P, Tan P. CoSLAM: collaborative visual SLAM in dynamic environments. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(2): 354–366

[26] Cai, Z., Chen, M., & Yang, L. (2011). Multi-source Information Fusion Augmented Reality Benefited Decision-making for Unmanned Aerial Vehicles. In 2011 6th IEEE Conference on Industrial Electronics and Applications (pp. 174–178)

[27] Qin T, Li P L, Shen S J. VINS-mono: A robust and versatile monocular visual-inertial state estimator. IEEE Transactions on Robotics, 2018, 34(4): 1004–1020

[28] Liu H M, Chen M Y, Zhang G F, Bao H J, Bao Y Z. ICE-BA: incremental, consistent and efficient bundle adjustment for visual-inertial SLAM. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, UT, USA, IEEE, 2018, 1974–1982

[29] https://www.myminifactory.com/object/3d-print-parrot-bebop-2-drone-templates-17700

[30] https://support.unity3d.com/hc/en-us/articles/360001252086-UNet-Deprecation-FAQ

[31] https://docs.unity3d.com/Manual/UNetStateSync.html

[32] https://developer.vuforia.com/target-manager

[33] https://library.vuforia.com/articles/Solution/Optimizing-Target-Detection-and-Tracking-Stability

[34] https://library.vuforia.com/content/vuforia-library/en/reference/unity/classVuforia_1_1CameraDevice.html

[35] https://library.vuforia.com/content/vuforia-library/en/articles/Solution/Advanced-Camera-API.html

[36] https://store.dji.com/product/tello-edu?vid=47091

[37] https://dl-cdn.ryzerobotics.com/downloads/Tello/ Tello%20Mission%20Pad%20User%20Guide.pdf

[38] https://www.gsmarena.com/xiaomi_redmi_4_(china)-8419.php

Appendix A: Targets (QR codes)

Board Target:

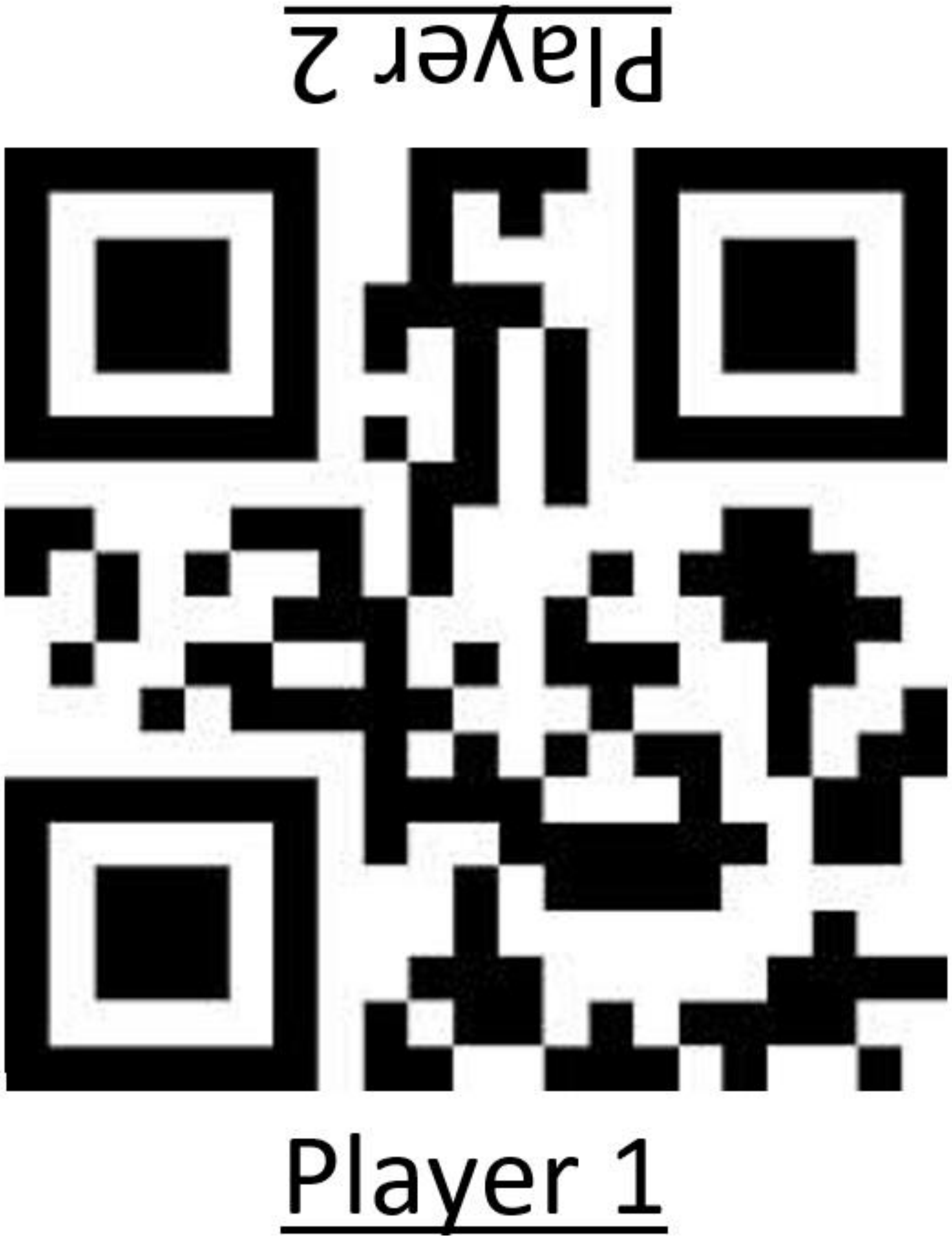
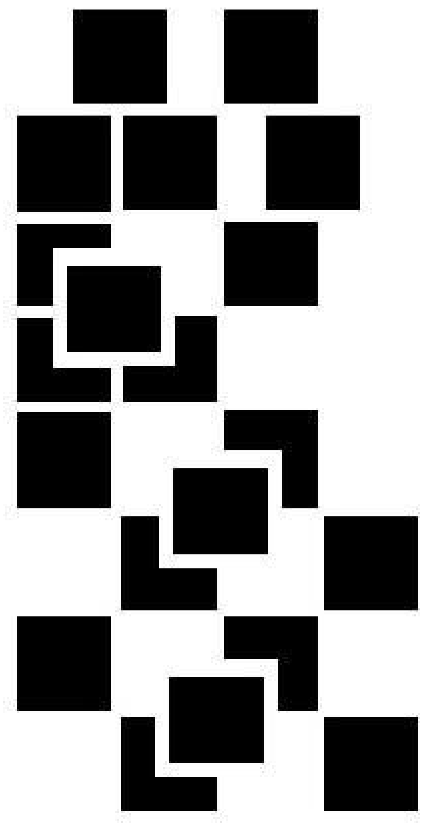


Figure 28: Board Target

Drone Targets:



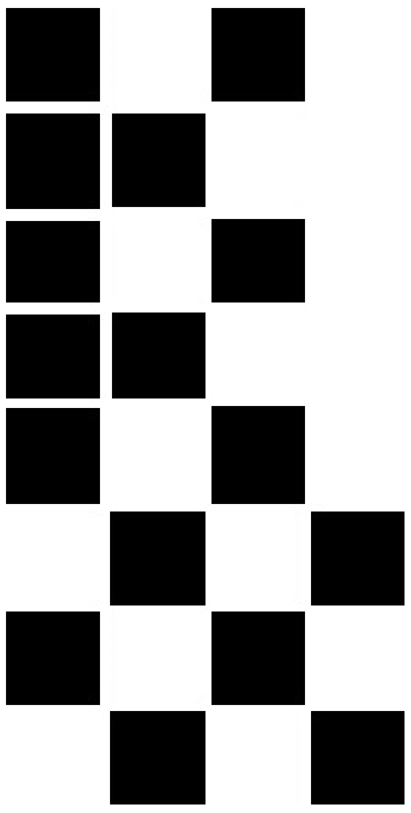


Figure 29: 5-star drone "Target" (left), 3-star drone "Target" (right)