Integration of Mobile Devices into a Floor-Based Game to Increase Player Dynamics

Andreas Friedl

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2015

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, September 28, 2015

Andreas Friedl

Contents

Preface

Through my studies and a lot of spare time spent on watching tutorials, reading books, doing research, talking to other developers and so forth, I was gifted with the opportunity to specialize in what I always wanted to do. Interactive and entertaining multimedia systems have been my passion since my childhood. Now, as a student who focuses on game development, I gained the ability to take an interesting idea and create something out of it more or less from scratch. This is something I am truly grateful for and that would not have been possible without my family, friends and colleagues. I want to thank those people for letting me follow my passion.

In the last couple of months I spent a lot of time in the *Deep Space*, physically and mentally. Walking around this multifunction room at the *Ars Electronica* museum in Linz and interacting with its applications is a stunning experience. However, for me the even greater experience is to explore the interaction possibilities for this setup and the search for new ways to impress its users. When designing an application for such an interesting environment, one needs to imagine how it is being used, how the people are perceiving it. That's when one as a designer enters the room mentally, envisions an idea, improves the design over and over again until, eventually, it is brought to perfection.

I really hope that people will have as much fun playing with *MoCo* as I had designing and creating the framework and its games.

Abstract

This thesis shows the diversity of *co-located floor-based* video games and elaborates on how the user interface in such games can be extended. A particular focus lies on how to give the player more control over the game and more ways to perform explicit actions during the game.

First a selection of *co-located* video games is described and some of its technical features are discussed. Furthermore, there are deeper insights into the field of *smart gaming*, a genre in video games which integrates so called *smart devices* (mobile multifunctional devices, such as smartphones and tablets) directly into the game play.

MoCo is introduced as a novel concept. It is the approach developed by the author to integrate mobile devices in *co-located* games and, thereby, on the one hand to extend the controls and on the other hand to offer a new game experience. In order to be able to rate the capability and efficiency of this system, a *game experience survey* with a selected group of users as well as an *expert heuristic evaluation* with focus on *usability* and *playability* were conducted.

Finally, the results of both evaluations show that *MoCo* is not fully matured yet and requires further improvement. However, the idea behind the concept has the potential to pave new paths in *co-located floor-based games*.

Kurzfassung

Diese Arbeit zeigt die Vielfältigkeit von *co-located* und *floor-based video games* auf, und nimmt sich der Thematik an, wie die Schnittstelle zum Spieler in solchen Systemen erweitert werden kann. Im Speziellen geht es darum, dem Spieler mehr Kontrolle im Spiel zu geben, also mehr Möglichkeiten explizite Aktionen durchführen zu können.

Dazu wird zuerst auf bestehende *co-located floor-based video games* eingegangen und Besonderheiten dieser Systeme werden aufgezeigt. Im Weiteren gibt es auch einen Einblick in das Feld des *smart gaming*, ein Genre im Videospielbereich, bei dem sogenannte *smart devices* (mobile Multifunktionsgeräte, wie etwa Smartphones und Tablets) ins Spiel integriert werden.

Als neuer Ansatz wird anschließend *MoCo* vorgestellt. Dabei handelt es sich um den Versuch des Autors mobile Geräte in *co-located video games* einzubinden und somit einerseits das Steuerungskonzept zu erweitern und dadurch ein neues Spielerlebnis zu kreieren. Um die Tauglichkeit dieses Systems bewerten zu können wurde einerseits ein *game experience* Fragebogen an eine ausgewählte Gruppe von Testspielern ausgehändigt, sowie eine *Expertenevaluation* mit der Hilfe von *Usability* und *Playability* Heuristiken durchgeführt.

Die Ergebnisse beider Forschungsmethoden zeigen deutlich, dass *MoCo* zwar – wie für einen Prototyp üblich – noch Kinderkrankheiten aufweist, aber die Idee dahinter durchaus das notwendige Potenzial mitbringt, um in *co-located floor-based video games* neue Wege beschreiten zu können.

Chapter 1

Introduction

This thesis describes *floor-based* game installations and discusses interaction models for these game systems. *Smart gaming* in the context of *co-located* games is described and finally, a novel way which aims to increase player dynamics via the use of mobile devices as an interface to the game is explained in detail.

1.1 Motivation

The basis for this thesis was formed through the development of five floor-based multiplayer games. Together, these games form the *Game Changer* suite [11]. It was first shown at the *Ars Electronica Festival 2014* in Linz and afterwards became a permanent exhibition in the *Ars Electronica* museum. The main idea behind the project was to provide a collection of games which make use of different styles of cooperative and competitive play in a co-located setting. For the system architecture a layered approach was chosen, mostly for maintainability and convenience reasons. This basically means the games make use of an underlying framework which takes care of most common controls and parameters. The layered architecture offered valuable flexibility during the development, however, soon, it became clear that certain boundaries were reached with the system.

The five games offer diverse experiences and play-styles but in the end in *Game Changer* the players are very limited in the ways they can interact with the system and with each other. These limitations are described in more detail in Chapter 2.2. Also, for the developers the design process was challenging because of these constraints. To give the players as well as the game designers more freedom, the author wants to explore the field of player dynamics in floor-based co-located games even further.

1.2 Problem Statement

The main problem the developers were confronted with, during the development of *Game Changer*, is that players are limited in the way they can interact with the system. In *Game Changer* the players control their virtual avatar with their physical bodies. A tracking system based on multiple laser rangers spread over the floor computes the positions of the players in the playing area. This system provides fast response times and is capable of tracking up to 25 people at the same time. But even though position data in form of Cartesian coordinates and all sorts of additional data (velocity, estimated position, approximated orientation) are transmitted from the tracking system to the game server, in the end, the players only have their body position to interact with the system.

To work around this constraint in *Game Changer* certain player actions are automated: In *Swarm Defender* the players' spaceships shoot laser beams automatically in a short timed interval because there is no obvious way to let a player decide when to trigger a shot. Another approach to facilitate user interaction is to make intensive use of collision detection: In games which involve virtual objects which should be collected or even dragged the most evident way to achieve this is by letting players simply run over them. The virtual collision between the object and the player avatar would then cause the object to be collected or parented onto the avatar. This approach works quite well in games in which objects can only be used in one particular way. In *Tower Of Power*, for example, there is a particular zone in which players collect blocks and another zone in which those blocks are dropped automatically. Unfortunately, as soon as the player should have a choice on what action he or she actually wants to perform (e.g. with a virtual object) there is no obvious or user-friendly way to handle this.

These are the major problems encountered during the development of *Game Changer*. Exploring solutions to these issues was the main motivation for the author to write this thesis. There are, however, more challenges in the design of games for floor-based co-located systems as can be seen in the Chapters 2 and 3.

1.3 Objective

The aim of this thesis is to explore alternative and additional control mechanics in floor-based co-located games with focus on inclusion of personal mobile devices. Control models in existing co-located game setups are analyzed and advantages as well as limitations of those systems are discussed. From this basis, a novel system for increasing player dynamics in floor-based multiplayer games is inferred. Subsequently, a prototypic implementation of a novel system, called *MoCo*, which aims to integrate most of the findings,

is described.

1.4 Document Structure

Chapter 2 of this document deals with basic principles on the matter of floor-based co-located play. In the first section, a selection of existing floor-based co-located game systems is described. There are systems for commercial usage and systems built and used for research. A close look is taken at the *Deep Space* installation as it is also used by *MoCo*. The second section elaborates on game design challenges in co-located play. Besides some general aspects, there are also specific issues discussed concerning the development process of *Game Changer*. Pro and contra of the found issues are described and in some cases solutions are proposed.

Chapter 3 goes into detail on utilizing mobile devices in video games as non-primary devices. The first section describes the three levels of interaction barriers and suggests ways to overcome them. In the second section, problems, limitations and other requirements that come with smart gaming in a co-located setting are discussed. Opportunities and benefits of having a personal smart device in floor-based games are also part of that section.

In Chapter 4 *MoCo*, a framework for including mobile devices into floor-based co-located multiplayer games is introduced. In the first section, the requirements of a novel system which offers more dynamics to the players are described. The basic idea behind *MoCo* as well as its features are also explained. *MoCo*'s system architecture is described in the second section. It explains the network layer, how *MoCo* is implemented as an extension of the *Unity TUIO/Pharus Tracking Client*, the server and client role of the application, the process of identifying players, sensor calibration and how different game views are handled by *MoCo*.

Chapter 5 is devoted to evaluating the game experience and usability of *MoCo*. The first section describes the two demo games based on *MoCo*, *Lazor Arena* and *Lazor Lab*, which were used for both evaluations. The second section elaborates on the game experience survey which was conducted in the course of this thesis. The third section describes the expert heuristic evaluation of *MoCo* which also was carried out in order to locate usability issues and potential for optimisation of the system. For both evaluations the methods as well as how the evaluations were conducted are described in detail and the results are listed and discussed.

Chapter 6, finally, concludes this thesis. The first section sheds some more light on the results of the evaluations. Future plans and potential improvements for *MoCo* are described in the second section. The last section gives a brief overview of the current state of the art and further prospects for the field of *co-located* games.

Chapter 2

Floor-Based Co-Located Play

Video games have always appeared in various social contexts. One of them is co-located play[1]. Over the next few paragraphs, this section elaborates on some of the rather vague terms. Most of them are by no means new and also exist beyond research, still they might require additional definition in the context of co-located play.

Of course, not just video games but all kinds of interaction can take place in co-located settings. As a basic rule of thumb, one can say that all systems which establish an environment in which multiple users are interacting at the same time in the same physical space with the system and with each other can be considered as co-located systems. Through the co-presence of its users, various interpersonal interactions are facilitated, such as full-duplex communication, non-verbal communication, eye gaze and face-to-face interaction. Co-located systems come in particularly handy for group work and group play. Tasks like decision-making, coordinating, planning and designing usually benefit from the diversity and rapidness in communication [7, 35].

Co-located play is an area in which *sociability* naturally occurs. It refers to the joy emerging from the social component when being together with other human beings. In video games this phenomenon occurs when people come together (virtually/online or physically/locally) in order to play games together or against each other [16, 36]. Systems which incorporate co-located play particularly in social spaces and with full-body input are also part of what can be described as *social immersive media* [37]. In the field of video games, this type of play used to be practised mostly in the living room via video game consoles or during LAN parties. When online gaming became more popular and technically mature in the early 2000s, local multiplayer went through a rather lean period of time. However, co-

[1]It is noteworthy that the terms *co-located*, *colocated* and *collocated* often are used in the same context and therefore can be handled as synonym, at least in the game research field.

located play recovered well just a few years later, around 2006 when the Nintendo Wii console was released and motion sensor games became more and more popular. Nowadays, there is a broad variety of local video game systems available which utilize various kinds of controls and input mechanics and take also place in more diverse spaces such as urban and public space games.

The field of co-located games and interactive installations has been the focus of research for several years. Still there are areas which are mostly unexplored. One example are *floor-based* games. Unfortunately, the term *floor-based* has not gained enough popularity yet to get its own clear definition. In this thesis the term is used to describe installations in which one or more users are able to interact with the system by moving their physical body inside a defined space on a planar surface. In most cases, the users' position is tracked by a camera-array and an associated computer vision algorithm [15]. There are, however, other ways to achieve body tracking, e.g. through ground-based laser rangers [27] or floor embedded pressure plates [24, 38]. Usually, all these systems have in common that they project virtual avatars at the positions of the players in an interactive environment. The projection on the floor is either done via projectors or floor embedded displays.

According to Ballagas [1], interaction with large displays can be placed in three different types of spaces:

Personal space: An interactive installation in the personal space can only be used by a single user and usually is located in a private setting, such as at home or in an office building. As for co-located play, the personal space setting is not suitable by definition.

Semi-public space: An installation is semi-public if it is situated in a controlled access environment. Only a limited amount of people can interact, often collaboratively. For co-located play, this space is suited perfectly as it provides the opportunity to regulate access to the application and allows intervening directly if interaction is unclear.

Public space: Large open areas with usually high pedestrian traffic and/or extended wait times such as train stations are also suited well for co-located play.

The rest of this chapter provides an overview of the field and elaborates on selected existing systems which offer co-located play in floor-based games. The main focus lies on installations in public or semi-public environments. Interaction limitations and design challenges when working with these systems are described as well.

2.1 Co-Located Game Systems

There is a broad range of floor-based co-located game installations. They vary in technical regards as well as in the social environment they are placed in. Some of these installations are quite experimental or have been built for research purposes whereas some others are even commercially available. In this section insights and some details on selected systems are given.

2.1.1 Commercial Products

Body tracking has existed for quite some time and it has reached a state in which the technology is sophisticated enough to be commercially viable. Today, there are many commercial floor-based gaming systems available. To stay within the scope of this thesis, three of these systems are described briefly.

GroundFX[2] is a visual display system which uses a video camera to capture body movements. A software-side computer vision algorithm then tracks individual players and detects motion-based gestures. Unfortunately, as *GroundFX* is a commercial product, there is no documentation publicly available which explains how the tracking exactly works and what gestures it is capable of recognizing. An example usage of the system can be seen in Fig. 2.1 (a).

MotionMagix[3] is a ceiling mounted unit which uses either an integrated 3D sensor or an optional external 2D motion sensor, both of which are not described in detail. The projection is either handled by the device itself or an external projector is utilized. Also multi-user point tracking, hand tracking, foot tracking, full skeleton tracking, silhouette tracking and depth tracking are supported. Same as with *GroundFX*, there is no detailed documentation of the used technology for *MotionMagix* publicly available. *MotionMagix* is pictured in Fig. 2.1 (b).

EyePlay Floor[4] is an interactive floor system which is especially aimed at kids. It is not publicly documented how the system works, but from pictures on the company website, one can assume that the system also uses vision-based image processing to detect body-movement within a defined area. The system is pictured in Fig. 2.1 (c).

From the product folders as well as the pictures and videos on the websites it can be concluded that these systems are not trying to break new grounds in co-located interaction but instead focus on a solid user-experience and game

[2]More about *GroundFX®* by GestureTek can be found online: http://www.gesturetek.com/gesturefx/productsolutions_groundfx.php

[3]More about *MotionMagix™* by TouchMagix can be found online: http://www.touchmagix.com/interactive-floor-interactive-wall

[4]More about *EyePlay™ Floor* by EyeClick can be found online: http://eyeplay.info/floor/

(a)

(b)

(c)

Figure 2.1: Diverse commercial floor-based co-located game systems: *GroundFX* by GestureTek (a), *MotionMagix* by TouchMagix (b), *EyePlay Floor* by EyeClick (c).

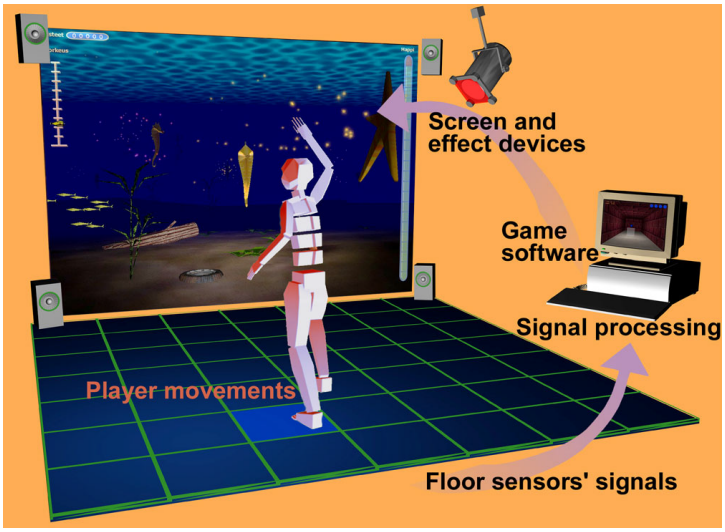


Figure 2.2: The *Lumetila* installation uses pressure plates in the floor to track the players' movement.

play which is easy to grasp. As all three example products apparently use vision-based motion tracking it can be assumed that this technology offers a convincing cost-benefit ratio and is mature enough for commercial usage.

2.1.2 Research Projects

In the research field of co-located game systems there are some interesting and also quite exotic approaches to implement body tracking. The following installations are either research projects or publicly documented through scientific papers.

One of the rather early projects is the *Lumetila* installation from 2001 [24] (Fig. 2.2). It was built with the goal in mind to create an interactive natural environment with a shared experience between the players. The system tracks people via an array of sensors arranged below a tiled floor and responds to changes in pressure. This approach provides a simple way to detect movement without requiring the players to wear any kind of additional equipment (e.g. helmets or gloves with motion sensors) but also makes it hard to locate still standing people. Additionally, the system is capable of recognizing outliers in the pressure measurement (e.g. rapid up-and-down movements of arms) and interprets them as gestures. However, it is not documented how accurate this gesture detection works.

An application specifically made for the *Lumetila* installation is the *Nautilus* project [38]. It is a collaborative multiplayer game in which the players form the crew of a diving bell. It is their mission to rescue a dolphin at the bottom of a lake. The players control the diving bell with synchronized body

movements and movements on the floor. The game requires them to avoid obstacles and to collect bonus items on their way down. The paper which describes this installation does not mention it being showcased in public but rather being evaluated by experts and selected groups of end users in a closed environment. It can only be assumed that the technology was not mature enough at that point to be used for a broader audience.

Lumetila is the only co-located gaming installation found during research for this thesis which utilizes pressure plates. The technology is likely not the most suitable for tracking movement of multiple persons. The approach seems rather cumbersome as it requires a very specific hardware setup, spatial preconditions and thus does not easily scale.

iGameFloor [15] is an interactive floor platform with vision-based tracking and bottom projection. It is built 3 meters into the physical floor and is covered with a projection surface. Four web-cams film the surface bottom-up in order to provide fine-grained tracking of limb positions. A lightweight version of the *iGameFloor* installation can also run with a single ceiling mounted camera which provides coarse-grained body tracking. The first approach, however, is preferred as it eliminates the problem of perspective and does a better job at separating multiple users' contours from each other. In both setups the cameras are managed by a computer which processes the video signal through a vision software. Since the application is tracking limbs, it is possible to detect gestures like a three-step walk-sequence.

A completely different approach is being used for *YOUPLAY*. It is also an interactive installation with wall and floor projection. However, the users are equipped with wearable interfaces. Players get to wear helmets with built-in infra-red markers which are tracked by a camera mounted to the ceiling. Additionally, the helmets have accelerometers in order to detect shaking or slight movements and help to track the players' orientation. As *YOUPLAY* is set up as a story driven 30-minute live entertainment show, each player engages in one of ten predefined character roles. Some of the characters have even more additional equipment, such as a toy gun or a butterfly net which also have hardware augmentations (accelerometer, Arduino Fio, XBee, and Lithium ion battery) installed (Fig. 2.3). Thereby, the players are able to trigger certain actions in the game with their tools [17].

2.1.3 Deep Space

A setup which is quite unique due to its large dimensions is the *Deep Space*[5] at the *Ars Electronica* museum in Linz. It is a multifunction room which is used for presentations, interactive art installations, games and more. Six industrial laser rangers[6] are mounted across the room covering the floor and thereby comprise an extensive tracking system. This architecture is capable

[5] http://www.aec.at/center/ausstellungen/deep-space/

[6] Hokuyo UTM-30-LX1 LiDAR (http://www.hokuyo-aut.jp/)

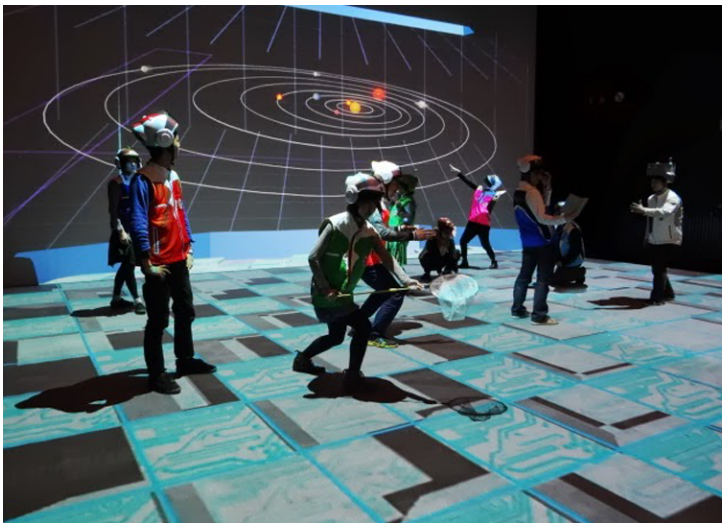


Figure 2.3: In *YOUPLAY* the players wear helmets with integrated motion sensors. Some players even have additional equipment such as butterfly net.

of tracking the position of approximately 25 people at the same time and has three major benefits in comparison to e.g. image-processing of a video stream:

- Laser rangers are not affected by brightness and thus can be even used in very dark or bright environments.
- Laser rangers can be placed directly on the floor which means that there is no need to consider a camera which might block the projector's field of view or perceive a distorted view of the scenery [15].
- The problem of privacy[7] is avoided, which is a rather sensitive topic [41].

A computer processes all the tracking data and broadcasts it over the local network. The network broadcast can be configured to use both the open *TUIO* protocol [18] via UDP/OSC.NET as well as the proprietary *Pharus* protocol [27] via either TCP or UDP. This way any application which implements the client side of one of the two available protocols can utilize the tracking data, even at the same time.

The application's video signal is then projected both onto the floor in up to 8K[8] resolution via four 4K projectors mounted to the ceiling and on the rear wall also in up to 8K via four additional 4K projectors. Both the floor and wall projection measure 16 by 9 meters. The projection on the wall can be configured to either mirror the floor projection or to show additional

[7]Some tracking systems (especially in public space and especially image based technologies) are faced with privacy concerns as it's possible to identify people.

[8]8K resolution or Full Ultra HD (FUHD) refers to total image dimensions of 7680×4320 pixels.

content.

There are several game applications available for the *Deep Space* tracking environment. Some of them are described in the following section.

Game Changer

Game Changer[9] is a semi-public interactive installation which was specifically made for the *Ars Electronica Deep Space*. It is a collection of five different multiplayer games. The focus of the project is to explore cooperative and competitive game-play in a co-located setting and was initially developed for public showcasing during the *Ars Electronica Festival*[10] in 2014. Since then it was shown several times at other public events and in early 2015 *Game Changer* was taken into the regular exhibition program of the museum.

In *Game Changer* each person inside the tracked area is assigned a visual avatar (depending on the current game). The avatar gets projected onto the player's position and follows the person as he or she moves in the room. The video output is additionally mirrored on the wall. This helps orienting oneself in the digital environment. Each of the five games functions only by means of players' physical presence and movements.

The game *Fish Feast* (Fig. 2.4 (c)) is a slightly adapted version of playing *tag* (also known as *it* or *tip you're it*). In *Fish Feast* the players are represented as fish in an ocean. The goal for each player is to consume as many non-player fish as possible and thereby grow in size. The largest player fish becomes the king (signified by the crown on its head) and from there on can also devour other player fish and eliminate them from the game. A player fish automatically eats another fish (turning it into an inactive fish bone) by approaching it and overlapping their collision boundary circles. As the boundary circles grow with the size of the fish it gets easier and faster over time to eat other fish. That is why there is a computer controlled electric eel from time to time wandering through the ocean which attempts to take the leading player fish down a bit. The game ends once there is only a single fish left alive. In the development of *Fish Feast* one of the major challenges was to avoid physical collisions between players. Section 2.2 elaborates more on this particular design issue. Also an important aspect was to balance the game in a way that players who would perform rather bad in the early phase of the game would still have a chance to win the game.

Another game part of *Game Changer* is *Swarm Defender* (Fig. 2.4 (f)). It is a collaborative game in which the players must work together to defend earth from extraterrestrial invaders. In this game each player is controlling a spaceship which automatically fires green laser beams. Invaders appear in three variations: green, blue and red. Green invaders are destroyed through

[9] http://game-changer.at

[10] http://www.aec.at/c/deepspace-gamechanger

a single player's green beam. In order to destroy a blue or red invader two or three players have to join forces by standing next to each other. A group of two players will fire a blue beam and a group of three players will release a powerful red laser. The dynamic in the game requires the players to coordinate among themselves in order to match their fire behaviour to the onslaught. The game ends either in a positive way once all invaders are destroyed or in a negative way once enough invaders manage to break through the players' defence line and deal damage to earth.

There are three more games in *Game Changer*: *Tower Of Power* (Fig. 2.4 (d)) draws its inspiration from the game classics *Tetris* and *Bomberman*, combining them into a team-based building race. By carrying *Tetris*-shaped blocks from the energy source in the center to the home base, each team can construct their own tower that eventually connects to the source and draws on its energy. *Beelzeball* (Fig. 2.4 (b)) channels the energy of competitive ball sports such as football or squash with computer game elements inspired by classic games like *Arkanoid* and *Snake*. But rather than being a team sport, the focus here is entirely on competition. And then there is *Fluridus* (Fig. 2.4 (e)), a game which combines competitive and cooperative play in a space setting. Players collect asteroid fields of varying sizes and colors that provide energy for each team's mother ship. After every of the above games the *Game Changer* application switches to a game selection (Fig. 2.4 (a)). In here players can choose between any of the five games by simply standing on the desired game window. Once the timer has expired the game with the most people standing on it (votes) will be loaded. That way, the game selection is a collective and democratic process.

Limelight

Limelight[11] [16] is another game specifically made for the *Deep Space* and works a little bit differently than *GameChanger*. The game is projected on the wall solely as the game's aesthetic matches the traditional two-dimensional platformer genre. A single player controls a wizard ("Limus") via a gamepad controller. The wizard's task is to find three gems in a dark cellar in order to escape. As the player's vision is very limited through darkness (darkness in the game, not actual darkness in the real environment) he or she is reliant on the help of the players in the tracking area of the *Deep Space*. Each of these players controls a Will-O-Wisp ("Lumee") which illuminates its surroundings. This asynchronous cooperative gameplay makes the game already very interesting. However, there is another mechanic which involves the audience: People can make monsters appear in the game world by sending a text via SMS or email to a specified phone number or email address. Players can even give their monsters ("SMorcS") names.

[11] http://pie.fh-hagenberg.at/projekte/student-projects/limelight/

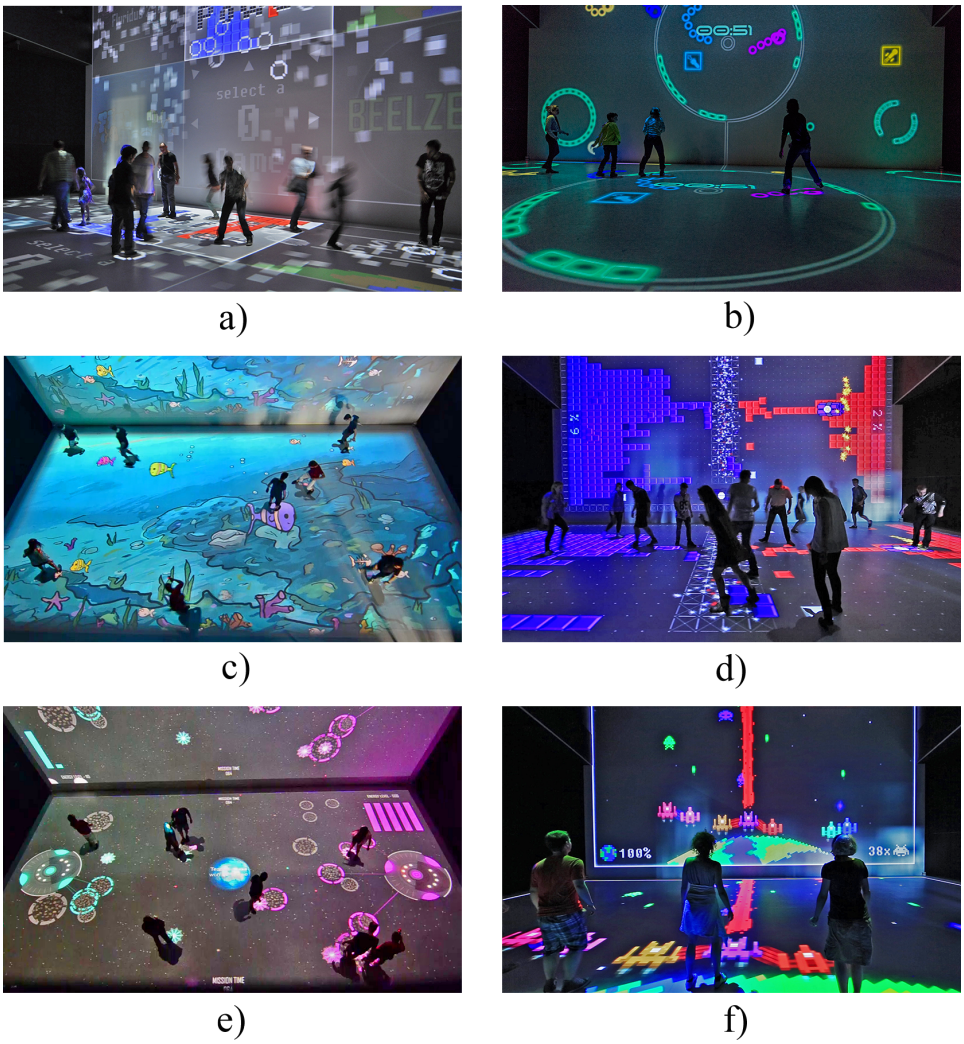


Figure 2.4: All applications of the *Game Changer* suite: game selection screen (a), *Beelzeball* (b), *Fish Feast* (c), *Tower Of Power* (d), *Fluridus* (e), *Swarm Defender* (f).

Concluding, it can be said that there is a broad variety in technology used among co-located gaming systems. However, a problem which all presented systems have in common is that the game design is highly depending on the system's capabilities and restrictions. In the following section game design challenges which occur in this context are described.

2.2 Game Design Challenges

Designing games for co-located game settings is a rather hard task as various unique factors come into play. This section handles special limitations and, consequentially, lists some requirements for game design for co-located games in order to work around these limitations.

Instant drop-in and drop-out: As for public and semi-public installations, it is a necessity to allow passers-by to easily join as well as leave the game. Any obstacle in this process will hinder the experience not just for the joining or leaving player but also for the other players. In an optimal case there is no player limit and people can join the game immediately, play as long as they want and then leave any time without affecting the game for the others. However, of course there are games which require a limited player access. In *Fish Feast*, for example, all players who join an already started round (after a certain time) they will instantly switch to the inactive fish bone state. This ensures that a round of *Fish Feast* cannot artificially be prolonged by already eliminated players rejoining the game. However, this also has the drawback that a player who leaves the tracking area by accident not just loses his game progress but also is permanently eliminated from the current round. This issue partially is the reason why most of the games in *Game Changer* avoid having player progress or complex player states. There are more details on this issue in one of the following paragraphs (cf. *anonymity of the player* 2.2).

Player proximity: In a co-located setting usually there is no way around coming physically close to other players. In fast paced games like *Fish Feast* or *Tower Of Power* the co-located setting introduces important concerns of player proximity in the form of real physical collisions. Players might get distracted of the many game events or flashy visual effects and forget to watch their local surroundings. Particularly games in which players run a lot and often take quick turns in direction are affected by this risk. There are measures that can be taken to reduce the chance of physical collisions, however, it does not change the fact that this is a rather exotic and peculiar limitation for the game design process of co-located games. Some the precautions which can be maintained are to slow down the game pace, to increase the size of the players' bounding boxes and to try to avoid distracting players' vision in general.

Inhibition to approach strangers: Another problem which arises in public space is that some people experience social inhibition when they are playing together with people they do not know. Games which requires them to approach strangers make them feel uncomfortable and often they react

restrained. In the worst case this might even lead players to quit the game. Similar to the fact that one can not expect every passers-by in public space to be experienced with video games, one also should not assume that every potential player is as extroverted and sociable as one would want them to be. Also for this issue there are some workarounds: The game design could consider the players' respect of other players. This would mean that a game does not force its players to closely interact with others. There could be passive player roles in the game which allow them to perform tasks which require indirect interaction rather than direct interaction with other players. When observing people playing the *Game Changer* games it can be seem that it is a good practice to not force players to harm each other. Concluding, collaboration usually works better than competitiveness if the players do not know each other that well.

Perspective: Another issue that arises especially in large scaled floor-based games is perspective. When interacting with large interactive displays users tend to start struggling between getting close enough to the display to interact with it, and getting away from the display far enough to put the whole picture into their field of vision [1]. In floor-based games it is usually not possible to step away from the display as the player is required to stay within in a certain area in which the tracking takes place. Also, as a logical consequence, the larger the display is dimensioned the harder it is to keep an overview of what is going on in the game.

Game Changer benefits from the option of the *Deep Space* to not just project the display onto the floor but also to mirror it onto the wall. This way the player can immediately see what is happening in his local surroundings by looking at the floor around his feet and look up to the wall at any time to see what is going on in the far away corners of the playing field. However, when there are many players in the game it can get difficult in the heat of the moment to find oneself's own avatar on the wall projection. Some kind of additional (unique) visual identifier on the public display (such as the name or a photo of the player) could help the player to find his or her avatar more quickly.

Lack of physical barriers: As floor-based games usually track their players' body position things get complicated when there are certain areas in the game which should not be accessible to players. In the real world it is not possible to prevent people from moving freely in the tracking area. Placing obstacles in the real world is often not an option as it is a danger for players. Co-located games already requires players to deal with a rather high cognitive load. Also obstacles in the playing area can interfere with the tracking technology's vision (may it be laser rangers or image based computer vision).

An approach to work around this issue is to cut the connection between the player's movement and his or her avatar. For example, a wall in the digital world can stop the avatar even though the player can continue his movement in the real world. But this method has a major drawback: Immersion in floor-based games significantly benefits from the direct mapping of the player's position to his or her avatar. Breaking this connection would drastically affect immersion.

Anonymity of the player: The next design challenge is also quite unique to public space games. The players in public co-located games are usually anonymous from a technical perspective. This means that the game is not capable of differentiating between players as it does not identify them. Thus, the game has no memory about a player's history or experience with the game. This leads to the problem that as soon as a game introduces player states, points or some other metric which allows any kind of progress for a player this limitation can quickly become annoying.

In *Game Changer* once a person steps out of the tracking area he or she automatically drops out of the game. Once the same person steps into the game again he or she will start as a new player as the game has no way to determine if the person already was in the game. But this issue of course does not only affect intentional re-registration. In the *Deep Space* the system in some rare cases can lose a player's tracking identity. This happens due to various reasons: occlusion, close proximity to other players or jumping. Usually the player automatically gets a new tracking identity assigned, but nevertheless his or her game progress so far is lost.

Explicit player actions: In co-located floor-based games the users' basic interaction is done through the movement of the players' bodies. The body tracking is then usually mapped to the player's position in the game. In some cases the tracking system is also capable of recognizing certain body gestures or even movement patterns. This motion gestures then can be used to trigger certain actions in the game. Unfortunately, there are some drawbacks to these motion based controls: Even with very sophisticated gesture tracking there still is the problem that gestures might get interpreted wrongly or not recognized at all. As gestures usually need to be performed very precisely the error rate could get problematic in hectic or action driven games. Also motion gestures need to be learned and trained before they can be used and it usually takes the players a while to internalize them. Furthermore, some floor-based environments might not even offer adequate possibilities to track gestures. The *Pharus* tracking system in the *Deep Space* for example is capable of recognizing crowd movement patterns [27] such as linear movement, circular movement and flocking. However, pattern recognition for single persons is not supported as for now.

The games part of the *Game Changer* suite work around this limitation by triggering certain user actions when the players' bounding boxes overlap specific areas or objects in the game. This method provides satisfactory results but does only work in specific cases. The actions players can trigger, for example, always depend on their immediate surroundings (defined areas in the game): A player cannot decide when to use a collected power-up in *Beelzeball* (cf. Section 2.1.3), instead the power-up gets activated automatically in the moment the player steps on it. Also the timing of triggering actions that way is rather imprecise due to spatial distance between player and trigger area, inertia while rapid body movement and possible movement delay. Not having the possibility to allow players to explicitly trigger actions is in the author's opinion the a major limitation in the process of designing games.

Individual player feedback: As floor-based games don't utilize individual input and output interfaces for every player, it is difficult to provide individual feedback to a specific player. For example, in *Fish Feast* it happens from time to time that a player's fish gets devoured but he or she continues playing just as if nothing had happened. This goes on until eventually the player notices that he or she has been eliminated from the game. There are sound and visual effects whenever a player fish gets devoured but especially in hectic situations and also in situations with many players involved these indicators get easily missed. Audio feedback in particular is ambiguous as it cannot be directed towards a specific player but instead everyone will hear it. So a player might hear the sound effect but cannot tell what it was caused by and which player is affected. This problem also occurs in scenarios in which the system loses track of a player's position and the he or she suddenly doesn't have a tracking identity anymore.

Player exhaustion: There is another design challenge which can often be seen in motion controlled games: floor-based games can easily cause physical exhaustion as usually extensive body movement is required. This problem can be clearly observed in games like *FishFeast* and *Tower Of Power* (cf. Section 2.1.3). A rather obvious workaround to the players limited physical endurance is to keep play durations short.

Distraction through immersion: An interesting phenomenon was discovered in *Limelight* when players were playing as "Lumees". Especially children were distracted a lot—probably by the engaging visual looks of their avatars—and did rarely focus on the actual gameplay. Instead they were playing with their Lumees and watching them move on the wall. Some of the players were even trying to touch their avatars on the wall projection [16]. The visual looks of the avatar as well as the responsiveness to the players

motion fosters the relationship between the player and his or her avatar. This immersive experience apparently can in some rare cases negatively affect the player's behaviour as it distracts from his or her actual task in the game.

Concluding, it can be said that there are a lot of unique aspects in floor-based co-located video games. Some these aspects, such as player proximity, the lack of physical barriers and the lack of explicit player actions can be worked around in a more or less satisfactory manner. However, they require a certain amount of inventiveness. For some other issues, such as individual player feedback or the general anonymity of players there seems to be no smooth way to overcome them without some kind of extension to the overall system. In Chapter 4 such an extension to the co-located game system in the *Deep Space* is presented.

Chapter 3

Second Screens in Co-Located Settings

Providing an additional display in video games is not really new or innovative any more. At least on consoles and PC it has been done already more or less efficiently by some of the biggest gaming companies, mostly for extending the user experience or providing additional content. The augmentation of digital experiences through one or more additional displays is called *second screen*. This concept has been applied in various branches of digital media. But so far, the use cases of second screen applications are mostly set in a private space of the user, usually the living room. While in passive media, in which the user can only consume content from the system, second screens are often used to provide (optional) additional content (e.g. trivia) to the content (e.g. movie) of the main media channel (main screen). However, in interactive media such as video games a second screen offers a much broader spectrum of use cases [12].

An exciting approach is to utilize mobile devices in this scenario. So called *smart devices* are usually laid out for being operated one handed and have a focus on multimedia and communication (touch screen, microphone, speakers, wireless network/internet connection, Bluetooth, Near Field Communication, infrared sensor). They are equipped with various sensors to determine the device's position (compass/GPS, barometer), its orientation (accelerometer, gyroscope) and its local surrounding environment (proximity sensor, ambient light sensor, cameras). Typical smart devices are smartphones, phablets and tablets [2, 32] but also convertibles or detachables (lightweight laptops which basically can be transformed into tablets). Obviously these extremely versatile and feature-rich hardware opens a whole new horizon of interaction possibilities.

Smart gaming as defined by Emmerich [12] is a subcategory of second screen gaming. It is characterized by the usage of smart devices as game controllers and their private screens in addition to a separate public display

(e.g. a TV or a projector) which is utilized as a main screen. In other words, smart gaming is the combination of private smart devices with co-located play in public space. Naturally, lots of problems and requirements but also opportunities arise in such a scenario. This Chapter elaborates on how to overcome interaction barriers as well as problems and opportunities of smart gaming.

3.1 Cross-Device Interaction Barriers

Social inhibition (the fear of being embarrassed in public) [4, 25, 33] and uncertainty about the interaction possibilities [29] are the main problems interactive installations in public space are confronted with. In 2014 Cheung [6] took a closer look on the combination of large public displays and personal mobile devices and found out that there are massive differences when it comes to people overcoming their inhibition to interact with these systems. Therefrom, it is stated that social inhibition is just one factor of many which can prevent people from interacting with the installation. Furthermore, the potential of using private mobile devices as a bridge between the user and the public screen is clearly emphasized. To foster this connection and to remove psychological hurdles in the best possible way, some more interaction barriers are defined and subsequently a broad spectrum of possible solutions is proposed which covers these issues.

When Cheung defined the following interaction barriers they weren't framed in any application specific context. However, the approaches presented in the following are perfectly applicable to games.

3.1.1 Three Levels of Interaction Barriers in Public Spaces

According to Cheung there are three different levels on which interaction barriers can occur [6]. These three levels overlap partially, as can be seen in Fig. 3.1.

Attraction level: The first level is the attraction level. It involves all interaction aspects during the first phase of contact and recognition between the user and the public installation. *Display blindness*, for example, is the phenomenon when the user can not identify the public display as such. This usually happens when the installation is not attracting the attention or interest of passers-by. Another barrier on the attraction level is *interaction blindness*. It occurs when the user does not realize that the installation is interactive or that it is meant to be interacted with by anyone. Lastly, there is *cross-device blindness*, which means that it is not clear to the user that his personal mobile device can and should be used to interact with the installation.

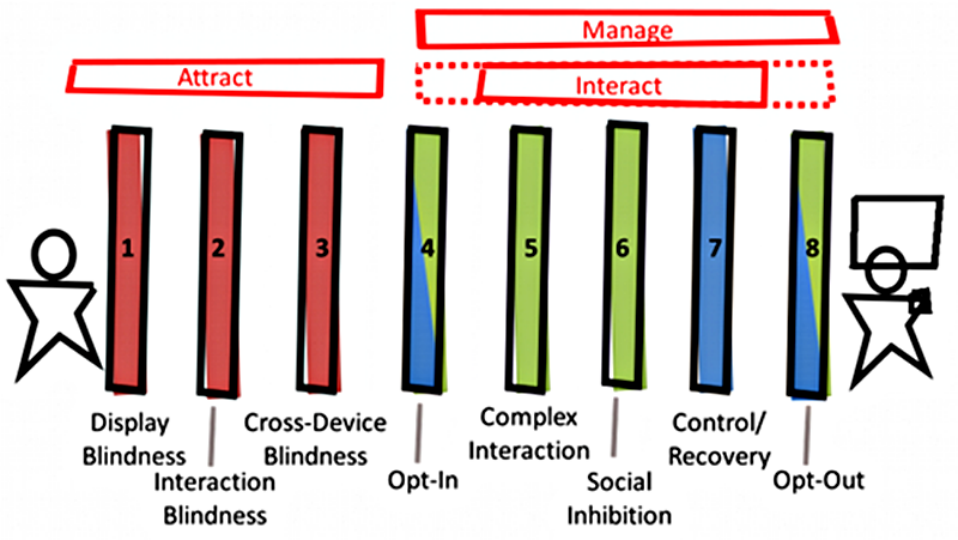


Figure 3.1: The three levels of interaction barriers in the order they usually appear according to Cheung [6].

Interaction level: The second level is the interaction level. It contains aspects which are connected to the understanding and usage of the mobile devices in combination with the public installation. There is *complex interaction*, which is a barrier that occurs when the interaction with the mobile device and the public display is not easy to learn or to understand. And there is *social inhibition*, a phenomenon which occurs when users do not feel comfortable and are afraid of failure or looking foolish while interacting.

Management level: Finally, the third level is the management level. It describes aspects of the connection between the mobile devices and the public display.

Inadequate opt-in/opt-out can cause frustration, for example, when it is not easy to engage and withdraw from interacting with the system. Connecting and disconnecting the mobile device from the public display does require a significant effort or causes frustration. At the same time mistrust might be generated if the system performs sensible actions on the user's private device (connecting to the system, downloading third party software and similar actions), especially if this happens automatically without explicit permission.

Also *missing control/recovery* can be fatal e.g. when interruptions are not handled properly or there is no easy and fast way to restore the previous state. As mobile devices are primarily stated as communication devices, incoming calls, messages or notifications might cause the interaction with the public space system to pause or, even worse, to stop [12].

3.1.2 Overcoming Interaction Barriers

For the above listed issues Cheung made a collection of approaches to solve them [6]. The following list contains proposed solutions to minimize or avoid these barriers. Once again, the previous division into three levels is being used.

Attraction level: Reacting to the presence of the users can help to overcome *display blindness*. The display could call attention to the system by playing automated animations or audio signals. In case it is equipped with sensors to track passers-by it can react to changes in its environment and show appropriate content and visual cues on the public display [6, 26].

To prevent *interaction blindness* a good way would be to show clear instructions directly on the public display. They can be animated to attract the users attention, however, the user must not be overwhelmed by too fancy animations or too much information all at once. A step-by-step-like approach in which the user is guided through the instructions and thus only gets to see the information he or she just needs is preferred [6, 40]. Passers-by are also attracted by music and sound effects. It's important, though, to not push it too hard since this might have the opposite effect. Also in most public spaces (e.g. shopping malls or train stations) insonification might not be an option due to local restrictions. If it's not possible to show these instructions on the public display there should at least be some signs or instruction labels be put next to the public display [6, 30]. In some cases it might also be an option to assign an on-site instructor directly to the installation. This person can then teach passers-by directly how to use the system. The author's experience with *Game Changer* has shown that this approach is very well received and usually works best.

Another way to help overcoming *interaction and cross-device blindness* is to provide signifiers. If the system uses sensors or eye-catching technology instead of hiding it it is a good idea to rather expose it. This is often understood as a sign of interactivity. To indicate that the system is used with private mobile devices a symbol (pictogram), photo or even a video of someone using his or her phone to interact with the display can be shown on the public display. Also, commonly known indicators like a QR code is often a clear indicator for interactivity [6]. Another important phenomenon which can be utilized is the *honeypot effect* [4]: Once someone starts interacting with the system, others will start watching and their inhibition threshold to join will be reduced significantly.

Interaction level: To reduce *interaction complexity* providing immediate interaction is a key strategy. The less steps the users are required to perform in order to connect their devices to the system the better. For example, a (shortened) URL or a QR code which redirects to the actual application are

preferred solutions [6]. Ideally, the user does not have to install an external application on the private device at all. The connection could be established through a web-app in the device's internet browser. Other examples on how this can be achieved even without requiring an internet connection are *BlueTone* [9] (by transmitting predefined dual-tone multi-frequency sounds via Bluetooth), *e-Campus* [8] (by manipulating the Bluetooth device name of the private mobile device) and *Blinkenlights*[1] [9] (by calling a phone number and pressing the numeric keypad to perform predefined actions).

Also, it is important to convey possible input modalities. Mobile *smart devices* are per definition (cf. Section 3) equipped with multiple sensors, and thus offer various ways to exchange input and data [2]. It is recommended to make use of the devices capabilities as efficient as possible: Instead of limiting controls to one particular way multiple alternative control schemes should be offered. For example, with a smartphone an avatar can be navigated on the public display via touch gestures on the phone's screen but additionally the same can be achieved by utilizing the phone's gyroscope [6]. It is important, though, that users are not excluded from the system if a certain hardware feature is missing on their mobile device. Therefore, controls with rather exotic hardware requirements should be offered additionally. Also, it is noteworthy that with this approach the various controls types must be pointed out and explained additionally [6].

As already mentioned in the introduction of Section 3.1, *social inhibition* is usually seen as the main reason for people to avoid public installations. One key strategy to reduce this inhibition is to provide appropriate feedback during display interaction, device connection and connection failures [6], since uncertainty about the state of the system, uncertainty about what interaction is possible and fear of failure are usually the main causes of social embarrassment [4, 33].

To make the interaction more easily understandable to the users it is recommended to provide a fail-safe test environment, a kind of tutorial in which new users can learn how the system works fast and get a feeling of how it reacts to input before the actual interaction even has begun [6]. Googles *Super Sports Sync* game[2], for example, has such a training mode. Before the actual competition starts the players can try out the controls already in the lobby while they wait for other players to join (fig. 3.2). To highlight the connection between the mobile devices and the public part of the system synchronized feedback between the two is recommended [6]. By offering such kinds of hints and exploration aspects it should be more easy for the user to learn the utilized interaction modalities.

[1] http://blinkenlights.net/blinkenlights
[2] https://chrome.com/supersyncsports/

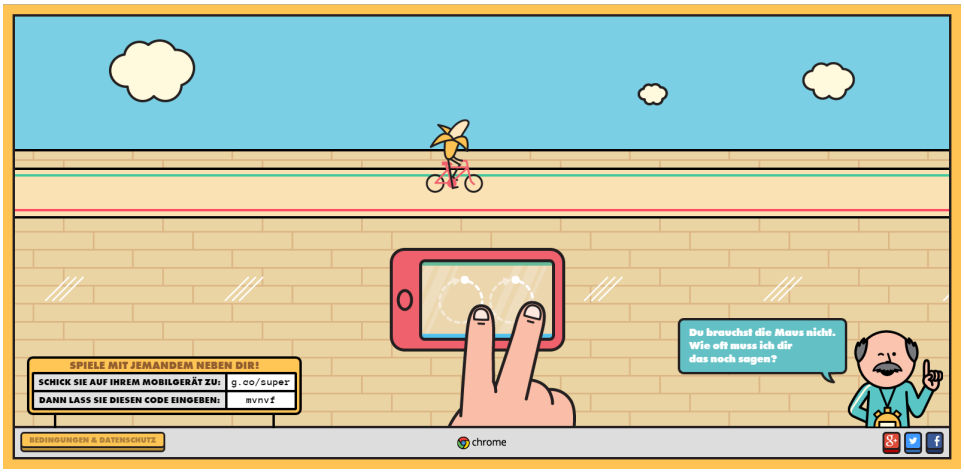


Figure 3.2: In Googles *Super Sports Sync* the player learns the controls before the actual game even starts. Also, there is synchronized visual feedback on the smartphone's display and the main display

Management level: To enable a flawless and smooth experience when connecting multiple mobile devices to the public display the necessary bandwidth should be minimized as far as possible by reducing the amount of messages sent in the network. Network traffic for data which usually requires a high-frequency update rate (e.g. orientation sensor data from the mobile devices) can be optimized by reducing the update rate and interpolating in between each update on the retrieving end (public display) [6]. Minimizing the bandwidth is especially important for games as responsiveness is a key factor for playability and user experience.

Keeping the user aware of the system state is especially important during the connection- and withdrawal-process. This can be achieved by conducting proper opt-in and opt-out modalities: The system must not download any data or connect to any system automatically without explicit permission by the user, otherwise the user might get insecure and mistrust the integrity of the system [14, 25]. For the same reasons the system must not expose personal data which could (from a technical point) rather easily be gathered from the users' mobile devices. Seeking the user's explicit permission for joining the system has also benefits when it comes to avoiding false positives. Again, wrongly interpreted user actions usually foster social inhibition [3] (e.g. a passer-by's device is immediately connected and shown on the large display even though the person just wanted to check out the web link which is shown on the display).

Another approach to prevent frustration on the management level is to support proper recovery & control mechanisms: The system should be able to handle the circumstance that some users might not explicitly sign off their

mobile devices but simply walk away. In such a case one solution would be to introduce a timer which signs off idle devices after a certain period of time [6]. In case of an interruption or accidental disconnection of a device the re-establishment of the previous connection state should be handled as user-friendly as possible [6, 12]: If some kind of user-authentication, network information, device-calibration or any other step linked with additional effort is required on connection, it should be possible to skip it by saving the particular user-specific parameters on the server. If this is not possible, at least sensible default values should be provided [6]. Also the application on the large display should handle connection issues properly: In a game, for example, when a player's device has connection issues his avatar on the big screen should freeze and get into some kind of idle state in which the player is safe of negative consequences (e.g. lose points) until the connection is re-established [6]. And last but not least, providing live drop-in and drop-out helps to make the experience more flawless: Users should be able to join and leave the system at any time without interrupting the experience for other users [12].

As shown above, there are quite a lot of interaction barriers in smart gaming which need to be addressed properly. However, smart gaming offers a lot more unique characteristics. Some of them can be hindering but some others can actually be beneficial if utilized correctly. The next section elaborates on some of these aspects.

3.2 Smart Gaming in Co-Located Settings

When it comes to designing games for co-located settings, there are already a lot of unique challenges. The utilization of smart devices in this context helps in many ways to overcome these limitations to some extent, but at the same time it brings in completely new aspects which require the designer's attention.

This section elaborates on further problems, limitations and specific requirements but also on benefits and rather unique opportunities which go hand in hand with smart gaming in co-located settings. Some strategies to overcome specific issues are presented as well.

Emmerich [12] already explored most of these design challenges and categorized them into three levels: cognition challenges, technology challenges and social challenges. Emmerich's classification is also used in the following.

3.2.1 Cognition Characteristics

Common video games connect their player through a single display to its digital environment. With second screen gaming the player suddenly is confronted with one or more additional displays. As a result the decision on

what to focus in which particular situation is imposed on the player. Thus, completely new cognitive challenges arise [12, 39].

Cognitive load: The cognitive load which comes with more than one screen hinders the player's processing of information [39]. An uncertainty about which screen holds the more important information may lead to frustration and might affect concentration. The game experience in general is put at risk. Fortunately, consistency and clarity in the user interface (UI) already diminishes a big part of this cognitive load. Less important or less frequently used UI elements often can be hidden in submenus to reduce the general amount of UI elements. Furthermore, context-relevant UI elements should be highlighted in order to help the player draw his or her attention to them quickly. And finally, special attention cues can signalize the player directly which display he or she should focus on (e.g. an icon in the shape of a smartphone which is shown on the public display, or, of course, the other way round) [12]. These cues do not have to be exclusively visual, also audio cues[3] or even haptic cues (vibration on the smart device) can be used to draw the players attention.

In large-display and floor-based games the players are often constantly moving around in the playing field. For example, most of the games in *Game Changer* involve fast-paced gameplay in which player proximity is a major concern [11]. For these kind of games the distraction through a second screen can even be dangerous as players not focusing on their surrounding might crash into each other. To reduce the risk of physical injuries the usage of the smart device should work intuitively and not require the players to focus on the screen for more than a few seconds. Thus, a drastically reduced UI and controls which can be operated blindly are preferred.

Asymmetric information distribution: Players having access to a private display offers game designers to distribute information across them in an asymmetric way. This means, while every player has access to the same base information (public display) each player can have additional individual information (private display) [12]. Moreover, not just information can be distributed unequally but also the players' abilities can differ. From a game design perspective this is a very interesting aspect since it allows relatively easy to emerge different player roles, and thus adds a very unique social component to the game. This diverse information distribution must, however, be explained to the players accordingly. It should be clear to the player which role he or she has and what actions can be performed based

[3]Some best practises on the use of audio feedback through small built-in speakers in gamepads or personal devices can be found here: http://gamasutra.com/blogs/RevDrBradleyMeyer/20150406/240483/Best_practices_using_the_PS4_Dualshock_controller_speaker.php

on the role. Otherwise confusion and frustration might arise quickly [12]. When it comes to asymmetric game design sophisticated balancing is required. Despite unequal abilities, resources, rules, or objectives each player should have roughly the same chance of winning. Often players are required to adapt their tactics and build alliances with other players. Maybe players even can change their role during a play session [13].

3.2.2 Technology Characteristics

As already stated above, smart devices come in various configurations. Not all players have the latest hardware on the market and therefore care should be taken when a broad range of devices is used in the same setting.

Hardware limitations: The explicit utilization of certain rather new hardware features can exclude a lot of older or low-budget devices. To serve a wide range of players it is essential to overcome this issue. Exotic hardware features like NFC (Near Field Communication), gyroscope, Bluetooth, fingerprint sensors and so forth, should be either avoided or made optionally (alternative controls) [6, 12]. Another way to handle this problem is to create specific roles for players with rather poorly equipped smart devices: e.g. in a team-based game in which motion sensors are mandatory for action-based tasks, a player without the required sensors could take the role of a commander and take over tasks like resources management or strategic planning.

Another hardware limitation which affects the game experience and thus requires a special game design is battery-life. Short durations for single play sessions are preferred so prevent smart devices running out of battery. Also, heavy and complex computations (especially graphics) should be avoided since they drain a lot of energy [12].

Balancing despite various device characteristics: The game design must work well with variations in hardware and ensure that certain characteristics of a personal device do not offer significant advantages. The *screen size*, for example, varies a lot between different device categories (smartphone - phablet - tablet). A larger screen must show the same content as a small one does to ensure everyone plays under the same conditions [12]. But again, players with a rather large screen on their private device could take a special role in a game, such as a commander or overseer. A clever usage of this approach is e.g. the commander mode[4] in *Battlefield 4*. It is not a co-located game per se but the point being made is that its concept could offer interesting gameplay for public space games.

[4] http://battlelog.battlefield.com/bf4/news/view/commander-mode-app-out-now-2/

Private devices versus shared devices: Device-based interaction with a large public display can also be done with publicly available mobile controls. By providing the necessary smart device directly on-site allows participation without specific hardware requirements for the users. This approach seems very promising on first thought, however, there are certain drawbacks which should be considered, such as maintenance, sanitation, multi-user simultaneity and the users' ability to spontaneously interact with the large display [1].

Furthermore, the aspects of individuality of smart gaming mentioned in the following (cf. Section 3.2.3) are not applicable when publicly shared devices are used instead of private ones.

3.2.3 Social Characteristics

The social aspects in public co-located settings vary a lot from common video games played on consoles or PC at home. The fact that they take place in public already has an effect on how people behave and how they interact with each other. The private smart devices also provide new ways to involve a broader range of player types in the game.

Audience participation: In public space naturally not every passers-by is a gamer. Besides that affinity for gaming, participation in public space games usually also requires extroversion to some extent. People who do not come up to these requirements may be afraid of performing badly or simply feel uncomfortable due to close contact with other players. These people might still be interested in the game but simply prefer watching over playing. Thus, a big portion of the possible target group of large-display games in public excludes itself already before it even gives the game a try. This issue, however, can be addressed once private mobile devices come into play. As already mentioned in the Sections 3.2.1 and 3.2.2 various player roles can be introduced [12]. For the above described rather passive player type a role with limited and simple controls might eventually attract them after all. A passive role in which the audience does not act as players per se but still can influence the course of the game is introduced in *Limelight* [16].

Individual user settings: Usually public space games are closed systems in terms of player progress. Every play round starts with all participants sharing the same basis as the players are anonymous. This is often due to avoiding a cumbersome authorization procedure. Smart devices, though, can be used to keep track of player's experience and progress. Points and other statistics can be stored in a private user profile on the smart device which is then synchronized with the game.

There are other benefits of this individuality aspect as well. The player can easily personalize his own gaming experience. The private mobile game

interface could provide the ability to customize the appearance or enter a name for the user's avatar which is then shown on the public display [5]. Furthermore, the player's private game interface can be altered manually or even dynamically to match personal preferences, habits or cognitive conditions [23].

Another aspect which profits through individual user statistics is the player diversity in a smart gaming setting. Since a lot of different people in terms of gender, age and expertise are involved, it is a design challenge to satisfy a wide range of concurrent demands [5, 12]. What Fullerton [13] describes as *balancing for skill* means for co-located public-space games that an algorithm can balance certain game parameters [5]. In contrast to matchmaking algorithms of common multiplayer PC or console games, which try to select players with equal skills and brings them together, this approach adjusts the overall difficulty and pace of the game to match the sum of experience of all connected players (profiles). This can happen before each game round or even during the game play.

As this chapter shows there are, compared to regular co-located gaming, some novel and interesting characteristics in smart gaming. Care has to be taken when designing applications for smart gaming. A lot of the design principles presented in this chapter were already applied in smart games and some more experimental applications (e.g. *First Strike* [25], *Multiplayer Breakout* [6], *Catch a Thief*, *Data Theft Algorithm*, *The Mole Rush* [12]). Also the framework presented in the next chapter makes use of some of these guidelines.

Chapter 4

MoCo: a Mobile Companion Framework for Floor-Based Games

While Chapter 2 clearly shows that there is a lot to consider when designing applications for floor-based co-located systems in public space, Chapter 3 points out the benefits and special considerations when bringing smart devices into public co-located installations. In this chapter an attempt is being made to define requirements to an extension of floor-based co-located games in order to overcome the mentioned limitations. This extension should bring new opportunities for players to interact with the game without interfering with existing modalities.

MoCo is such an extension. It is a framework for including personal mobile devices into floor-based games. It tries to work around most of the design challenges described in the previous chapters and to enable more versatile and attractive game design.

System overview: *MoCo* stands for *mobile companion* and is the author's approach to enhance floor-based games with personal smart devices. It has a client implementation (companion application for mobile devices) and a server implementation (co-located game in (semi-)public space). The server component is an extension to the TUIO/Pharus tracking receiver system used in the *Deep Space* setup. This means, while there is still a network client thread running which processes tracking information through TUIO/Pharus, it also additionally runs a network server which allows mobile devices to connect to the game server. The mobile component of *MoCo* is a fundamental structure and network interface for applications running on mobile devices (*Google Android* and *Apple iOS* at the moment). One job of the *MoCo* server is to handle the game server and all connected mobile devices. The other job *MoCo* does is to manage the references between the

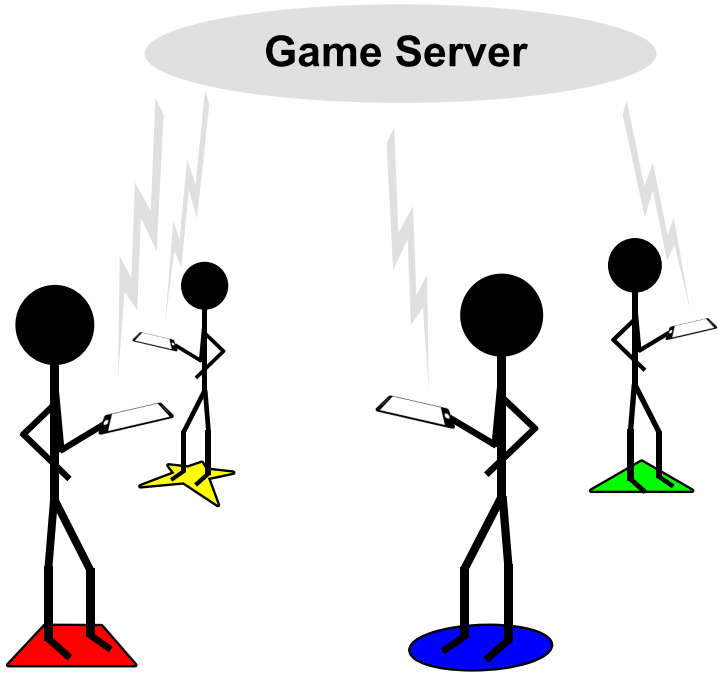


Figure 4.1: In *MoCo* the players' smart devices are connected to the central game server via WiFi. The identification avatars match the symbols in the identification view of the mobile application (fig. 4.4 (b)).

mobile devices and the corresponding tracking identity.

The idea of *MoCo* is that each player who has joined the game by physically stepping into the tracking area also holds a smart device in his or her hands which is connected via WiFi to the *MoCo* game server (as pictured in Fig. 4.1). Therefore the *MoCo* server/client interface provides the functionality to identify a player and assign it to his or her corresponding TUIO/Pharus tracking id. As soon as a player's smart device is connected to the game server and the player has been assigned the correct tracking id the mobile device acts as an extended UI to the game. The game still takes place on the public display (floor projection) and all the game logic runs on the server side, but the player receives personalized audio and tactile feedback through the handset when his or her avatar in the game is being confronted with any kind of interaction. This way the player's perception in the game is noticeably extended. But *MoCo* also works the other way round: the player's smart device is used to send commands from the player directly to the game. At the moment the system supports two types of commands: permanent transmission of sensor data (e.g. gyroscope) and input based commands such as pressing and releasing of a button. The *MoCo* client structure therefore suggests a simple design and supports basic UI elements

such as buttons. Any other more complex 2D or 3D interfaces are neither explicitly supported nor recommended, in order to keep the cognitive load as low as possible (this subject is also discussed later in Section 4.1.1).

In *MoCo* all the game logic happens on the server side, the machine which runs the actual game. This ensures that it's harder to manipulate the game by sending unexpected messages over the network. The server will simply ignore every message he does not understand. Also, each individual player state in the game is stored on the server side. Though, once a game is over, the player state (profile, score, etc.) gets synchronized also on the private mobile devices. This is necessary as the game server is not capable of storing this information between game sessions at the moment. The handling of user data might not be really matured yet but it also allows the users to check on their user profiles without being connected to the game server. However, the future development will most likely include an improvement of this architecture.

4.1 Characteristics of MoCo

In this section some characteristics of *MoCo* related to the matter of the previous chapters are being focused on. It focusses on requirements to the system as well as features of the system.

4.1.1 Requirements

The previous chapters elaborated a lot on problems and limitations. In this section requirements in order to overcome these issues are inferred and their implementation in *MoCo* are described. These requirements are used to underpin the concept on which *MoCo* is based on.

Seamless drop-in and drop-out: As previously described it is essential for public installations to allow *instant drop-in and drop-out*. It ensures that the usage barrier is kept as low as possible. In the design process of *MoCo* it was of great importance that the process of joining and leaving a game would work as easy as possible while still conducting opt-in and opt-out.

To connect the mobile companion app to the *MoCo* server the player has two options: When the player joins the game for the very first he or she has enters the IP address of the game server into the respective input field and join the game by pressing the manual connect button. Once the connection is established the IP address of the server gets cached in the mobile app. When connecting to the server the next time the last used IP address is proposed automatically. The second and obviously more user-friendly way is to simply press the auto connect button and let the client listen for a network broadcast message provided by server. Once the broadcast was received the client retrieves the server's IP address from the message and connects to the

game. This usually takes not more than three seconds but depends on the network traffic. In a setting in which the network is known and persistent the IP address could innately be stored directly in the mobile companion app's settings. Unfortunately, this is not often the case, so for new users the IP lookup or manual input is required.

Whenever the player leaves the physical tracking area of the game, as the tracking id gets removed, his or her mobile companion app automatically gets unassigned from its tracking record. The mobile app, however, is still connected to the game server so that the player quickly can rejoin the game without being required to reconnect to the game server. The player then only needs to re-identify his or her tracking id. In order to disconnect from the game server completely the player can do so by pressing the according button in the settings menu. As this option is usually not required often it is placed in the settings menu intentionally. This ensures that the player does not press the disconnect button by accident during regular game play. Of course once the game server shuts down or the mobile app is closed the client gets disconnected from the server.

Minimizing cognitive load: The additional screen of the smart device in a *MoCo* setting can easily lead to the described issues with increased cognitive load (cf. Section 3.2.1). Therefore some precautions have been made to minimize cognitive load.

The *MoCo* UI is being held clear and consistent: The mobile companion app is separated into different views. The main view is shown when the device is not connected to a *MoCo* server. It shows an input field for the player's in-game name, a big button which starts the auto connect process, another input field for the IP address of the server, and finally a small button which is used to connect manually to the game server via the provided IP. The player is usually not confronted with the main view during regular game play as it is only required to establish the connection to the game. Once the application is connected the identification view is shown. It shows four buttons, each with a specific color and geometrical shape. In this view the user can easily and quickly assign him or herself to one of the available tracking records in the game. A more detailed insight on this step is given in Section 4.2.2. The simple UI of this view ensures that the process of identification works intuitive and fluently. The next view the player gets confronted with is the calibration view (only in games which utilize data of the gyroscope sensor). Again, this view is kept very simple, there is only a single button which causes the gyroscope data to calibrate. This step is mandatory the first when connecting the first time and from then on the calibration offset is cached (more on the calibration can be read in Section 4.2.2).

After identification and calibration is done the app switches to the game

view. The game view depends on the current game running on the *MoCo* game server. All game views have a settings button in the top right corner of the screen. It is part of the overlay of the *MoCo* framework. The menu button opens a menu which holds some *MoCo* specific options, like re-identification, re-calibration and disconnecting from the server. Aside from the *MoCo* specific UI, the game view's UI is recommended to consist of not more than 3 big elements such as buttons. Buttons which trigger actions only in a specific context or situations are greyed out and only get interactable in these specific situations.

There are lot more design challenges discussed in the Sections 2.2 and 3.2 which obviously need to be addressed. However, issues such as *inhibition to approach strangers*, *player exhaustion*, *distraction through immersion* or even *hardware limitations* are not in the scope of *MoCo* and instead should be handled by the game system itself, the environment it is being set up in and the actual game play design.

4.1.2 Features

Some of the unique aspects of smart gaming are implemented in *MoCo* in a beneficial way. These features are the core of *MoCo* and the reasons why *MoCo* is an enhancement to regular floor-based co-located games.

Individual player feedback: The lack of individual player feedback can be problematic in co-located games (cf. Section 2.2). Fortunately, as smart devices are equipped with various hardware features such as a display, speakers and small vibration motors, they are a great opportunity to personalize feedback. *MoCo* makes use of all these features in order to increase immersion and to make the connection between a game event and the involved players clear.

As soon as the system loses track of a player's position (e.g. when he or she steps out of the playing field) the player's handset will vibrate to signalize that the identification was lost. This measurement ensures that a player immediately gets informed in such a situation and therefore prevents frustration. In other games without *MoCo* (e.g. in games part of the *Game Changer* suite) it can be observed that when players lose their tracking they usually are confused as they often notice it somewhat later.

While the just described feature is implemented in the *MoCo* foundation of the mobile application it is also possible for the game specific part of the mobile app to receive individual feedback from the game server. Vibration signals can be used to draw the player's attention from the surrounding environment to the private display. Audio signals can be used to emphasize player specific actions such as firing a gun. As only a specific player fires the gun and only another specific player is eventually hit by the bullet not

everyone needs to hear the according sound effects (only the shooting player's device would play the shooting effect and only the hit player's device would play the hit sound). Visual signals can be used to display player specific information on the player's private screen. For example, an action which needs to be performed by a single player over a certain amount of time could be shown as a progress bar on the private display. As this information is only of relevance to the performing player it would be distracting for other players if the progress bar was shown on the public display instead.

Explicit player actions: Virtual buttons in the UI of the mobile companion app allow players to trigger action in the game. The *MoCo* framework implements a communication interface which let's the player send commands through the companion app to the game server. These commands are tied to the player's device id. As the server side *MoCo* system knows to which tracking record a mobile device is coupled, these commands are player specific. In order to also allow user actions over time, two separate signals are sent per button press (on the button down event and on the button up event). The communication interface of the server side keeps track of the pressed buttons for each user. This approach ensures that the network bandwidth is not unnecessarily loaded but at the same time requires the usage of a network protocol which handles the loss of network packets properly. *MoCo* also handles the input from sensors like the gyroscope or accelerometer of connected smart devices. These signals are sent continuously in order to guarantee high precision. Section 3.1.2 suggests to reduce the update rate of sensor data and interpolate between the updates on the server side. During the development of *MoCo* and also in field tests with up to 10 connected smart devices it turned out that this step actually is not required and there was no measurable delay in the network when continuous sensor data transmission was enabled. For the further development of *MoCo* it is planned to perform more extensive tests with even more connected devices to confirm this assertion.

Player profile: As already mentioned earlier (cf. Section 2.2), the anonymity of players in common co-located games is difficult to handle when different player states are introduced in the game design. Most of the stated problems are handled by *MoCo*'s architecture. Because players need to identify themselves in the game by assigning their device to a tracking record, the system knows which player actually is in the game and which state he or she currently is in.

In Section 3.2.3 the benefits of player profiles are explained. *MoCo* implements a lightweight version of a player profile on the mobile client side. This profile currently stores the player's nickname and the score of individual games. Other previously mentioned data such as experience, time spent

in games, user settings or preferences are currently not implemented. Such metrics could and maybe will be added in future to allow an even more personalized gaming experience.

As *MoCo* is still a prototype there are some flaws in the implementation. Some features are not as generic as they could be. Unfortunately, the server and client side are not completely independent from each other yet and therefore only work with two specific example games (cf. Section 5.1). Also *MoCo* currently only works in the *Deep Space* setup on top of the existing tracking technology as the framework architecture is not encapsulated and adaptive enough. All these issues will probably be addressed in future releases of the framework. However, already at this stage *MoCo* demonstrates very clearly how smart devices can be utilized in co-located settings to increase player dynamics.

4.2 Implementation

While the previous section describes the overall architecture and the characteristics of *MoCo*, in this section the implementation is documented. As *MoCo* works as a layer on top of the tracking system of the *Deep Space*, the TUIO/Pharus tracking implementation is also described. Having a look at the greater picture helps to understand why certain things were implemented as they are.

4.2.1 Unity TUIO/Pharus Tracking Client

As already mentioned earlier (cf. Section 2.1.3) the *Deep Space* has a laser ranger tracking system installed that is capable of tracking the position, speed, movement direction as well as expected position for multiple people at the same time [27]. This data is broadcasted over the local network and can be subscribed by any other program in the same network.

The co-located games running in the *Deep Space* are made with *Unity3D*[1] (short: *Unity*), a free game engine which allows to create a wide range of interactive content and export it on various platforms. One characteristic of the *Unity* engine is that its application programming interface (API) is not thread safe. This means that all game related code needs to run in the *Unity* main thread and that code which runs in a different thread on the CPU cannot access any *Unity* specific code. However, as the scripting environment of *Unity* is plain and simple *.NET*[2] it is possible to create a

[1] https://unity3d.com/

[2] this is not entirely true as *Unity* actually uses C# in combination with *Mono* (http://www.mono-project.com) which is an open source implementation of Microsoft's *.NET Framework*.

new thread from the *Unity* main thread and let the two communicate indirectly (asynchronously). Both tracking client implementations (TUIO [18] and Pharus [27]) for *Unity* handle the data receiving the same way (therefore the following operational sequence is only explained for Pharus): In the *Unity* main thread (thread A in Fig. 4.2) there exists a `UnityPharusManager` (as a singleton) instance which instantiates and holds references to a `UnityPharusListener` object and a `UnityPharusEventProcessor` object. The `UnityPharusListener` instantiates and holds a reference to either a `UDPTransmissionClient` or a `TCPTransmissionClient` (the respective settings for each client type are read from an XML configuration file and passed on by the `UnityPharusManager` beforehand). The actual `TransmissionClient` then starts a new thread (thread B in Fig. 4.2) and listens for incoming data on a specified port. Once a tracking record is received the according event (`OnTrackNew`, `OnTrackUpdate`, `OnTrackLost`; depending on the life cycle state of the received track record) is called and the track record is passed to the `UnityPharusListener` which adds it to a queue container:

```
// UnityPharusListener enqueues a new track record
public void OnTrackNew (TrackRecord track)
{
  lock(m_lockObj)
  {
    m_eventQueue.Enqueue(new PharusEvent(track.state, track));
  }
}
```

This queue is then processed by the `UnityPharusEventProcessor`. The `UnityPharusEventProcessor` is also the component which informs other components of the tracking by using the event handler pattern. In the following code segment the `APharusPlayerManager` subscribes to these events:

```
// APharusPlayerManager subscribes the tracking events
if(UnityPharusManager.Instance != null)
{
  UnityPharusManager.Instance.EventProcessor.TrackAdded += OnTrackAdded;
  UnityPharusManager.Instance.EventProcessor.TrackUpdated +=
    OnTrackUpdated;
  UnityPharusManager.Instance.EventProcessor.TrackRemoved +=
    OnTrackRemoved;
}
```

The actual method which handles the queue in the `UnityPharusEventProcessor` is called from the update loop of the `UnityPharusManager`. It might be noteworthy that in this whole setup the `UnityPharusManager` is the only class which inherits from *Unity*'s `MonoBehaviour` class and therefore is attached to a gameobject in *Unity*'s scene graph and is being updated by the *Unity* game loop. This means the dequeuing is actually initiated from the *Unity* main thread (thread A in

Fig. 4.2) while the enqueuing from the second thread (thread B in Fig. 4.2). Therefore, every access to the queue is being made thread-safe by locking an empty object. Fig. 4.2 shows that the transmission APIs are decoupled from *Unity* as far as possible. In the following code segment the function which processes the event queue is shown:

```
// UnityPharusEventProcessor processes its event queue
public void Process()
{
  while (m_listener.EventQueue.Count > 0)
  {
    UnityPharusListener.PharusEvent aEvent;
    lock (m_listener.LockObj)
    {
      aEvent = m_listener.EventQueue.Dequeue();
    }
    switch (aEvent.PharusEventType)
    {
      case ETrackState.TS_NEW:
        if(TrackAdded != null) TrackAdded(this, new PharusEventTrackArgs
    (aEvent.TrackRecord));
        break;
      case ETrackState.TS_CONT:
        if(TrackUpdated != null) TrackUpdated(this, new
    PharusEventTrackArgs(aEvent.TrackRecord));
        break;
      case ETrackState.TS_OFF:
        if(TrackRemoved != null) TrackRemoved(this, new
    PharusEventTrackArgs(aEvent.TrackRecord));
        break;
    }
  }
}
```

Using the event handler pattern helps to loosen coupling in the code which is always a good thing. Also, it allows multiple modules to subscribe to the tracking events independently: The `APharusPlayerManager` receives events and uses them to manage the players in the game. A completely other class (e.g. an analytics tool) could also subscribe to these events and use them for other purposes without affecting the `APharusPlayerManager` at all.

`APharusPlayerManager` is an abstract class. The idea behind this is that each game that uses the tracking system implements its own version of the `APharusPlayerManager`. This ensures that the PlayerManager is always compatible to the `UnityPharusManager` and its following up classes. Thereby it's relatively easy to switch between multiple games within the same application (as e.g. in *Game Changer*). However, the `APharusPlayerManager` actually is not a required part of the tracking system.

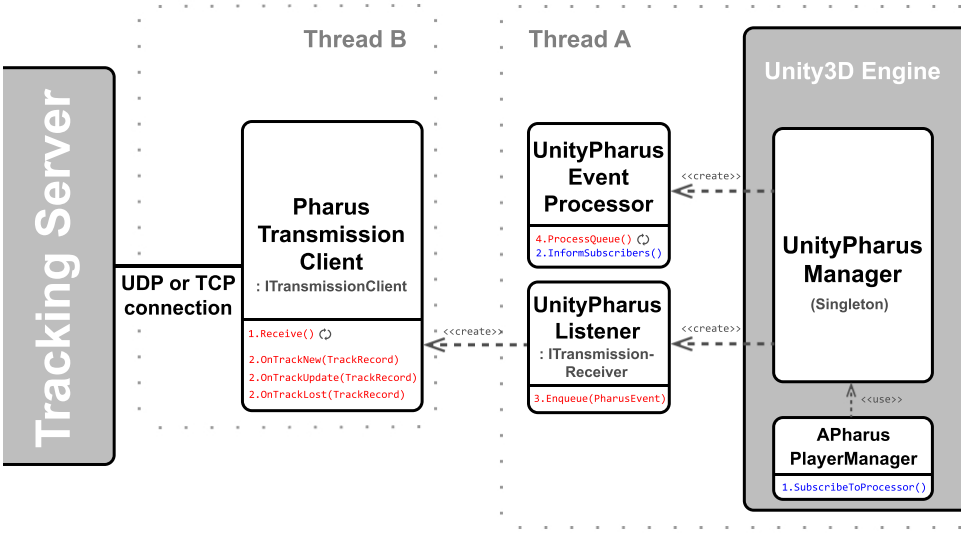


Figure 4.2: This diagram shows the most important components of the *Pharus* transmission architecture. Thread A is the *Unity* main thread while thread B is started by the `PharusTransmissionClient`.

4.2.2 MoCo Module Integration

Server side: On the server side one of the most important parts of what *MoCo* does is replacing the concept of the above described PlayerManager with the `PharusMOCOPlayerManager` (fig. 4.3): In a *MoCo* application when new tracking record is detected it does not create a player instance immediately because the track record is not identified as a player yet. Instead an anonymous entity is instantiated which moves with the actual player on the floor projection. In this state the entity does not have any affect on the game. Its visual look is either one of four colored shapes (red square, blue circle, green triangle, yellow star; pictured in Fig. 4.1), signalizing that the entity is ready and waiting for being claimed (identified) by a player, or it is displayed as a grey circle which means that none of the four colored shapes is currently available as other entities are already using them and are waiting for being identified. As soon as a player claims an entity it immediately switches to the actual game state and the game avatar is shown. The `PharusMOCOPlayerManager` keeps track of which entity is identified and which is not, therefore it updates a pending grey circle entity to a colored shape as soon as one is free again.

Another important job of *MoCo* is to run a network server to which the mobile smart devices can connect to via WiFi (cf. `MOCONetworkServer` in Fig. 4.3). As *Unity* comes with a built-in version of *RakNet*[3] there is not

[3] *RakNet* is a cross platform network communication solution developed by *Jenkins*

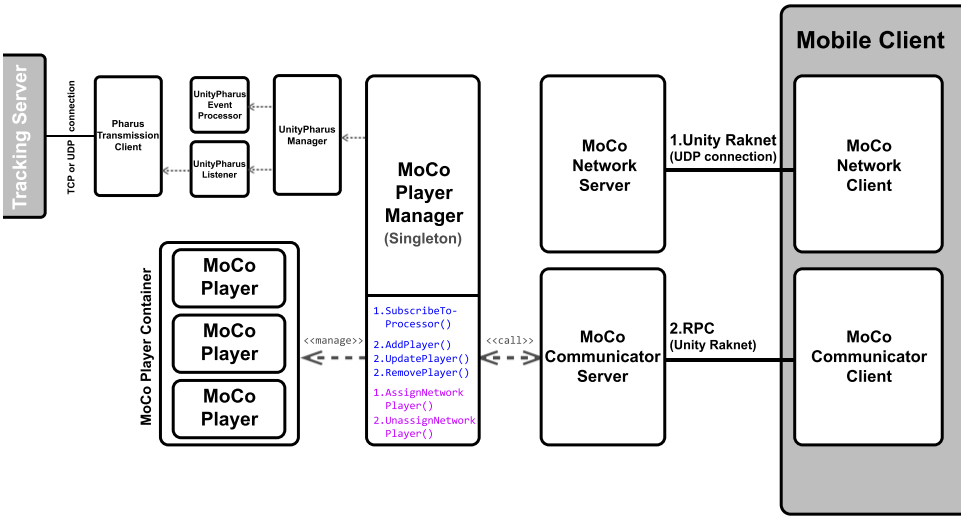


Figure 4.3: The `PharusMOCOPlayerManager` subscribes to the `UnityPharusEventProcessor` through the `UnityPharusManager`. The `MOCONetworkServer` and the `MOCONetworkClient` handle the initial network connection. All *MoCo* game and player specific communication is handled by the `MoCoCommunicator` which exists on the server and on the client side. Both implementations of the `MoCoCommunicator` know the respective opposite API and communicate via `RakNet` RPCs (remote procedure calls)

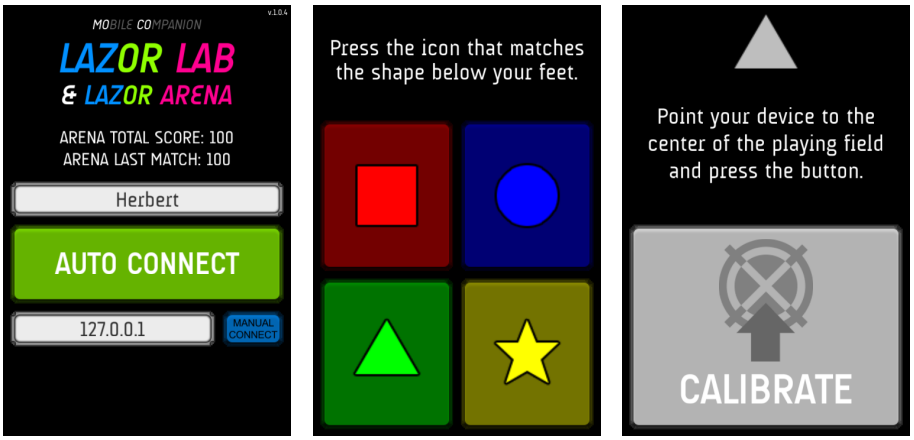
much additional code required in order to start and run the server. The server settings, such as used port, max clients and NAT Punchthrough[4] usage are hard coded in the current version. It is planned, however, to make *MoCo* more flexible in the future and read these settings also from an XML configuration file in the same way as the TUIO/Pharus network settings are read. As soon as the server is ready its IP address is shown on the public display and the players can connect their mobile smart device.

Client side: *Unity* was also used to make the mobile *MoCo* client as the engine supports also mobile platforms, allows fast development and offers a sophisticated build and test process. At the current stage *Android* and *iOS* are the only platforms the *MoCo* client runs on. Within the scope of future work additional mobile platforms such as *Windows 10 Mobile* might get supported as well.

The client application is divided into *MoCo* specific components and

Software (http://www.jenkinssoftware.com/) and was recently acquired by *Oculus VR* (https://www.oculus.com).

[4]NAT Punchthrough is a design pattern for overcoming connection issues when using NAT (network address translation) in settings in which multiple computers are using the same IP address (because they are connected to the network via a router).



(a) (b) (c)

Figure 4.4: The home view (a) is shown when the device is not connected to a game server, the identification view (b) is intentionally kept very simple, the gyroscope calibration view (c) is only shown at the very first connection and if gyroscope data is being used by the game.

game specific components in order to allow an easy adaptation for custom games or applications. The *MoCo* specific components handle the network connection, displaying the correct view as well as the management of user settings and sensor calibration data. The game specific components mainly handle the game UI and implement a game "communicator" script (more on that can be found in Section 4.2.2).

Network: *MoCo* uses *Unity*'s *RakNet* for connecting the client application to the server. *RakNet* in *Unity* builds on top of the UDP protocol which allows lightweight data transmission and due to *RakNet*'s implementation it adds a reliable layer in order to prevent data packet loss. The later feature comes especially handy when transmitting press and release commands of UI elements, as a missed command would get the two sides out of sync.

In order to connect to the game server, in the home view (fig. 4.4 (a)) the player either hits the "Auto Connect" button or enters the IP address of the server into the input field and then presses the "Manual Connect" button. In the home view the player also can enter a nickname which then is used by the game on the public display.

While connecting to the `MOCONetworkServer` is done by the `MOCONetworkClient`, the vast majority of the network communication is handled by so called "communicators". Each communicator component consists of a communicator script on each side (client and server). Both communicator classes are aware of each others signature and thus can exchange

data via remote procedure calls (RPCs[5]). Most of the communication between the two communicators makes use of the three-way-handshake pattern[6] to ensure that both sides can rely on messages being received. The *MoCo* relevant communication is handled by the `MOCOCommunicatorClient` on the client side and the `MOCOCommunicatorServer` on the server side. They mainly exchange data for identification, un-identifcation (unassigning a network player from a tracking record), sensor calibration and view changing. All game related data transmission is handled by a respective game communicator script pair.

Identification: The process of assigning a player's smart device to one of the tracking records is being referred to as "identification" in *MoCo*. As already mentioned earlier, as soon as a new player steps into the tracking area of the game system a visual entity is projected below his or her feet. For an unidentified player this entity has a colored geometrical shape (pictured in Fig. 4.1). Once the player connects his smart device to the game server, the identification view is shown on the mobile display (fig. 4.4 (b)). By pressing the according button on the player's private display, the game server knows that the network client of the smart device belongs to the tracking record with the matching shape. As there are only four available identification shapes, four unidentified players can identify themselves at the same time. Once a player identified him or herself the identification shape is being freed and gets available again. If there are more than four unidentified tracking records pending they will be visualized by a grey circle and change to one of the identification shapes as soon as one gets available.

There are of course other ways to solve the problem of assigning player to tracking records which might seem more sophisticated on first thought: Another way, for example, would be to project a unique QR code for each player in front of his physical position and then let the player scan it with his or her smart device's camera. However, that degree of complexity is not even required. During development and early tests with up to seven players the color and shape based system, in which the recognition part is simply being imposed to the player instead of a computer algorithm, seemed sufficient and worked very well. There were no situations in which players would have to wait longer than just a couple of seconds in the identification process. Anyhow, further evaluation has yet to be done to confirm this assertion.

[5]RPCs are *RakNet*s implementation of a straightforward API to send network messages between two or more known endpoints in a flexible and rather effortless way.

[6]The three-way-handshake usually is a sequence of three messages sent between two clients (A and B). The first message is a request from A to B. The second message is a request acknowledgement from B to A. The third message is a sync acknowledgement from A to B.

Calibration: *MoCo* allows the utilization of sensor data. One specific sensor, the gyroscope, allows to determine the orientation of the smart device the player is holding. *MoCo* offers particular support for the gyroscope sensor because it enables a lot new dynamics in player controls: The player can rotate his avatar according to his smart devices orientation. Before that in floor-based games it was only possible to have the avatar line up with the player's movement direction. By decoupling the orientation from the movement, aiming and targeting at specific objects does work much better and more intuitive.

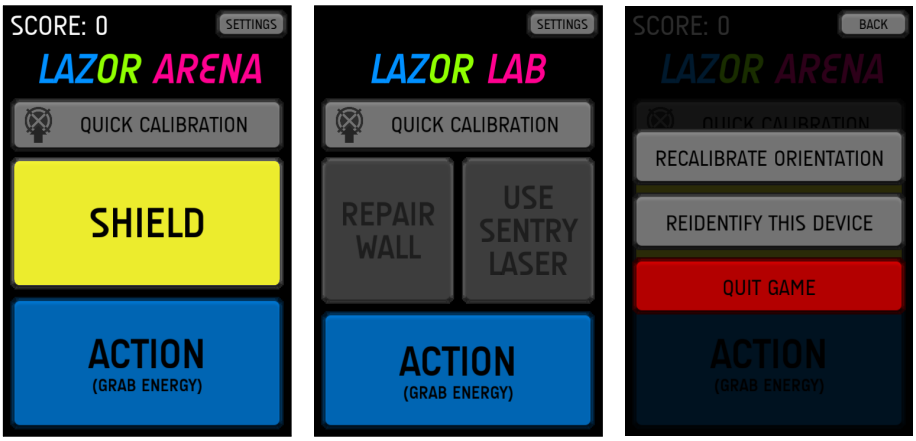
However, most sensor data of smart devices need to be calibrated before they can be used properly. For example, the gyroscope sensors of various smart phones (even of the same model) might deliver different results in the same situation. This is due to the way gyroscopes work[7]. In order to use this data properly, it needs to be equilibrated by adding a calibration offset value to the actual value the sensor returns. This offset value needs to be determined for every device. And unfortunately gyroscopes are also significantly prone to drift over time[8].

This is where *MoCo*s calibration comes into play. Once a player connects and identifies the first time in a game which utilizes gyroscope data, the calibration view (fig. 4.4 (c)) pops up on the private screen. The player now has to point his or her smart device to the calibration point in the physical room (which is usually the center of the tracking area) and press the calibration button. The game server then calculates the angular offset between the current sensors orientation to the calibration point and sends it back to the mobile client. From this point on the client adds the offset value to the gyroscope data before transmitting it to the server. The whole process literally just takes the blink of an eye. In order to correct drift over time, which might occur during a game session, the player can repeat the calibration process directly with the quick recalibration function from within the game view without interrupting the game play: By pressing the "Recalibrate Orientation" button (fig. 4.5 (a) and (b)) the whole calibration procedure is done instantly without calling the calibration view again. Of course the player, again, has to point the device to the calibration point while recalibrating.

The support for other sensor data (especially the accelerometer, microphone or camera) would hold a lot of potential but has not yet been implemented into *MoCo* at this stage. It most likely will be a part of the future development of the system.

[7] A gyroscope measures either changes in orientation or changes in rotational velocity but it is not able to determine absolute orientation data.

[8] There are solutions to this problem: So called inertial measurement units (IMU) use a range of different sensors (sensor fusion) to correct drift. However, only a small percentage of smart devices are capable of this technology and given the broad variety in smart devices (e.g. fragmentation of the Android operating system) probably neither should developers rely on this technology to be available.



(a) (b) (c)

Figure 4.5: The game view of *Lazor Arena* (a), the game view of *Lazor Lab* (b), the *MoCo* settings overlay (c) is available in every game view.

Game view: The game view on the mobile client shows interactive UI elements, such as buttons, and passive elements, such as labels. Labels on the private screen can hold player specific information, like personal score, but also general game information, like the remaining game time. However, it is essential that the game view is not overloaded with information to keep the users time focusing on the small screen to a minimum. Examples of game views are shown in Fig. 4.5 (a) and (b).

There is one constant *MoCo* layer which gets rendered on top of every game view. It has a small settings button on the top right corner. By pressing this button another overlay appears which holds *MoCo* specific options (fig. 4.5 (c)), such as re-identification, re-calibration and disconnecting from the game.

In general the game view is only shown when the client has been identified and sensor calibration has been done. If one of these preconditions is not fulfilled, the according view (identification or calibration) is shown but the game ID is still stored internally which means that the client knows which game view to load as soon as everything else is ready.

When a new client connects it receives the current game view ID from the game server and updates its view state internally. The game server is determinant which means that the client will stay in a game view until the game server tells it differently. Usually, when a game session is over the game server switches from one game to the next one. In this case the server informs its clients and they will update their game view accordingly. As *MoCo* handles all allocations between clients and tracking records, the players are not disconnected and also are not required to re-identify themselves.

On the game server, when a game is over, all *MoCo* relevant player in-

formation from the actual player object is being mapped onto a so called `MOCOPlayerDataImage` object. This is necessary as in *Unity* on a new scene load certain gameobjects in the scene graph are wiped. It does not make sense to keep the player objects as they hold data and references to assets which are all game specific. As soon as the new game scene is loaded, all player objects are restored with the new game specific data and their previously mapped *MoCo* data (such as track id, client id and identification state). This approach works very well in practice but unfortunately it also makes the code rather complex. Another more flexible approach would be to change the player object model to a component based model in which the *MoCo* specific part could persist over game changes and only game specific data would be wiped.

As mentioned, in its current state *MoCo* is a prototype and does not take full advantage of the potential of the technology yet. However, it is sophisticated enough for being used in an evaluation on how player dynamics can be increased in floor-based co-located video games and which problems arise when using approaches like *MoCo*.

Chapter 5

Evaluation

MoCo is a first approach in creating a framework for implementing smart devices into co-located games. It is based on a lot of research, trial and error and of course proven concepts found in literature. However, the implementation is new and not based on any code-wise foundation. As for every software it is important to make sure the structure of the application, its interface and its usage is not hindering the user experience. This is especially true when new software is built which does not benefit from iterations of previously versions or a sophisticated base framework. Typical usability problems found in games include menus that are cumbersome to use, displays whose meanings are not clear and controls that are difficult to learn.

For this reason two evaluations were conducted. A game experience questionnaire was filled out by people who played the two *MoCo* demo games *Lazor Lab* and *Lazor Arena* as well as *GameChanger* (in order to compare the two systems). Furthermore, a heuristic evaluation was carried out to find and document weaknesses in the usability of *MoCo* and the overall playability of the two demo games. The demo games are describe in the following Section 5.1, the evaluation methods and results in the sections thereafter.

5.1 Demo Games

MoCo is a framework which only works in conjunction with actual games. Therefore, two games based on *MoCo* were specifically developed for the *Deep Space* setting. In order to test a broad range of use-cases, the games focus on different play styles. Both games are combined in a single application so that the session administrator or an on-site supervisor can switch games without closing the application. Once a game is over, the other game will start automatically afterwards. Both games also have a pause mode which can be toggled by the supervisor. A paused game stops the overall progression and timer but still allows basic user interaction. This is especially practical to let new players learn the controls in a safe environment

Figure 5.1: Four people playing *Lazor Arena*: Three players have been identified (the colored circles are their game avatar), one player has not been identified yet and an identification shape is displayed (yellow star on the right). The magenta colored player has the shield activated.

before the actual game starts.

5.1.1 Lazor Arena

Lazor Arena (fig. 5.1) is a competitive battle arena game. Energy blocks spawn randomly from time to time across the whole playing field. There are three different types of energy: Lime, Cyan and Magenta. The players can collect up to three energy blocks from the floor by running over them. Collected energy blocks will slowly orbit around the player's avatar. Each player has his or her own gyroscope-controlled little arrow which acts as a pointer (fig. 5.2). By holding the smart device into a certain direction the arrow will point into the same direction (provided that the calibration has been conducted properly). When the arrow points to one of the energy blocks in the player's orbit, the player can grab it by pressing and holding the "Action" button on the personal screen. A player can hold an energy block as long as he or she wants but only one single block at the same time. As soon as the "Action" button is released the grabbed energy will transform into a laser beam and fire into the arrow's direction. This way the players can shoot at other players. The avatar (circle) of every player has one of the three energy colors, which color is decided randomly. A laser beam can hit a player if the color of the laser matches the color of that player. Once a player was hit his smart device vibrates to inform him or her about the hit and the avatar's color changes to one of the other two colors. However,

Figure 5.2: In *Lazor Arena* and *Lazor Lab* the smart device's orientation is mapped to the player's pointer (black arrow). The avatar moves with the player's physical position.

players can also block laser attacks by activating a time limited shield by pressing and holding the "Shield" button on the smart device. The shield is always aligned with the player's smart device. Eventually, a player scores points if he or she either hits another player with a matching energy beam or if a beam was blocked with the shield.

As the game is rather hectic and the players have to run a lot, one single round of *Lazor Arena* is only three minutes. Once the timer expires a ranking list with each player's name and score is shown on the public display. During the game the players can see their own current score also on their private screen.

5.1.2 Lazor Lab

Lazor Lab (fig. 5.3) is a collaborative game in which players have to defend their base from incoming enemies. There are a lot of similarities between *Lazor Arena* and *Lazor Lab*. For example, the players' avatars look similar despite the fact that they are grey all along. A player can still collect up to three energy blocks and also the grabbing and releasing of energy works the same way ("Action" button). However, players do not have a shield as it is not required. Energy blocks do not spawn randomly around the scene but instead there are five energy dispenser spread over the playing field

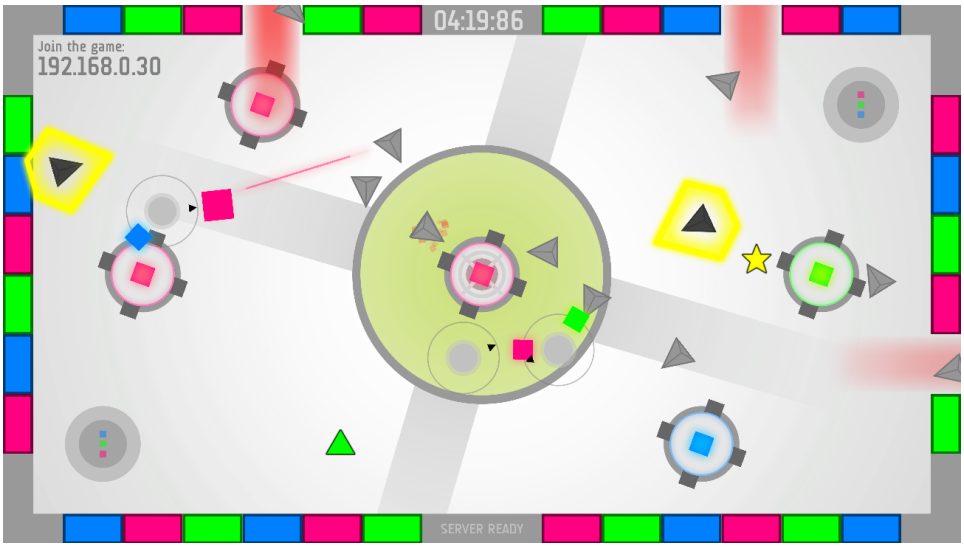


Figure 5.3: Five people playing *Lazor Lab*: Two of them have not been identified yet (green triangle and yellow star). Two players are exchanging energy cubes while the third player uses an energy cube to shoot at incoming enemies.

which always offer randomly one of the three energy types. In the center of the playing field is the core which needs to be defended by the players. The playing field is surrounded by a wall of 36 individual bricks. Each brick is colored in one of the three energy colors. From time to time individual bricks break and hostile polygons enter the playing field. Once they reached the core they slowly deal damage to it. The players have to shoot incoming polygons but also repair the wall in order to slow down the onslaught. A brick can only be re-built by a player who has three energy blocks of the same type. By pressing and holding the "Repair" button on the smart device while standing next to an open wall slot, will create a new brick. Only bricks of different types can stay next to each other. If a player e.g. builds a magenta brick directly next to another magenta brick both will explode. The game requires the players to coordinate themselves and handle their resources (energy) with care. Players can also exchange collected energy by grabbing a collected energy block and moving it on another player. There are also polygons with yellow shields. These enemies appear from time to time randomly in the scene and cannot be shot with a single energy beam. Instead, a player has to collect energy blocks of all three types in order to power one of the two stationary big laser guns in the corners of the playing field. By pressing and holding the "Use Sentry" button a player who meets the requirements can then shoot a powerful big laser across the scenery which destroys shielded enemies. Regular polygons are not affected by the

big laser but other players will lose their collected energy blocks when hit.

If the players manage to defend the core for six minutes they collaboratively win the game. If the core loses to many health points the players will lose altogether instead.

5.2 Game Experience Survey

The first evaluation method is a survey handed out to players. The aim of it was to find flaws in the design of the system and the games and to get an overall feeling on how the system is perceived.

5.2.1 Method

The questionnaire holds eleven questions. The first six focus on the game feel and experience the users perceived during playing *Lazor Lab* and *Lazor Arena*. Questions no. 7 to 11 are aimed to explore the relation between *GameChanger* (pure body controls) and *MoCo* supported games (body controls plus enhanced mobile interface). In order to reach the best results possible, the survey was conducted with the players directly on-site right after they finished the game session. It includes the following questions:

1. When grabbing a block it felt like I really grabbed it physically.
2. When shooting a laser beam it felt good.
3. I liked the way I could interact with objects.
4. I liked the way I could interact with other players.
5. The audio feedback on the smartphone improved the overall experience.
6. The haptic feedback (vibration) on the smartphone improved the overall experience.
7. In *Lazor Arena / Lazor Lab* I had more control over the game as in *Game Changer*.
8. In *Lazor Arena / Lazor Lab* I was able to influence the game's outcome more as in *Game Changer*.
9. In *Lazor Arena / Lazor Lab* I received more feedback/response from the game as in *Game Changer*.
10. Collecting an energy block in *Lazor Lab* felt better than collecting a (tetris-)block in *Tower Of Power*.
11. Shooting laser beams in *Lazor Lab* felt better than shooting a laser beam in *Swarm Defender*.

Each question is accompanied with a five-level Likert-type scale which is to be ticked directly on paper:

1. Strongly disagree
2. Disagree
3. Neither agree nor disagree
4. Agree
5. Strongly agree

5.2.2 Results

Altogether, 28 people participated in the survey. Unfortunately, as this number is too small in order to five representative results, the following interpretation should be treated with caution. The results of the survey can be seen in Table 5.1. For each question there are some metrics which help to understand the results: The range is calculated by subtracting the lowest given value from the highest given value and shows the overall breadth of answers. The standard deviation shows how much the average answer deviates from the arithmetic mean of all answers; it is the square root of the variance. Finally, to weight the results of each answer, there is the total mean which is calculated from the arithmetic mean, the median and the mode.

Regarding Q1, although some users agreed that when grabbing a block it felt like grabbing it physically, the overall result does not clearly mirror this (arithmetic mean: 3.11). The standard deviation (1.23) for Q1 also shows that people were perceiving the feeling of grabbing blocks quite differently. Q2, Q3 and Q4 have very similar results. This means that players seem to like the way how shooting laser beams feels and how interacting with objects works. Especially Q4, with extremely small values for range (2) and standard deviation (0.82), shows that there were no users who did not like the interaction with other players. Some players had troubles with answering Q5 as they claimed they did not hear any audio from the smart device. This is most likely why some people did not answer the question or ticked it rather bad (strongly disagree). There are three possible reasons how this could happen: Either, audio was disabled on the device by accident through the player or the surrounding sounds and music of the game and environment were too loud so that players could not hear the smart device or the personal audio feedback from the device was not perceived as such but as regular game sound from the public installation. More interestingly, when looking at Q6, there is a rather high total mean (3.91). This is a clear but rather contrary result: personal feedback from the device—at least vibrating—is very well appreciated and improves the overall experience. The results of Q6 strengthen the theory that Q5 could have received higher results if volume balancing in the test setting would have been better. The results of Q7 only shows a slight tendency towards agreement. From the relatively broad range (4) and deviation (1.24) it can be concluded that regarding this questions there was an uncertainty among the players. Most likely people

	Range	Deviation	Ar. Mean	Median	Mode	Total Mean
Q1	4.00	1.23	3.11	3.00	2.00	**2.70**
Q2	3.00	0.98	4.00	4.00	4.00	**4.00**
Q3	3.00	0.77	3.93	4.00	4.00	**3.98**
Q4	2.00	0.82	4.00	4.00	4.00	**4.00**
Q5	4.00	1.32	3.31	3.00	3.00	**3.10**
Q6	4.00	1.10	3.74	4.00	4.00	**3.91**
Q7	4.00	1.24	3.67	4.00	4.00	**3.89**
Q8	3.00	1.08	3.81	4.00	5.00	**4.27**
Q9	3.00	0.96	3.96	4.00	5.00	**4.32**
Q10	3.00	0.97	4.09	4.00	5.00	**4.36**
Q11	3.00	1.20	4.17	5.00	5.00	**4.72**

Table 5.1: Results of the game experience survey.

found it hard to compare the five games of *GameChanger* with the two games *Lazor Lab* and *Lazor Arena*. However, Q8 aims in a similar direction and received surprisingly positive results (total mean: 4.27). Also Q9, which compares the *MoCo* games with *GameChanger* in terms of feedback from the games (the main focus of *MoCo*), received very similar positive results (total mean: 4.32). Q10 and Q11 aim for a direct comparison of two different implementations of two game mechanics: collecting blocks and shooting laser beams. According to the results both mechanics feel better in *Lazor Lab* or *Lazor Arena*, however, shooting laser beams (Q11) got better results than collecting blocks (Q10). Both actions are supported by audio feedback from the personal device and both do not trigger a vibration feedback. The better results for Q11 might come from the fact that with *MoCo* it is required to trigger the laser beam by the user directly (button press or release), which likely increases immersion and the feeling of being in control compared to the automatic shooting mode in *Swarm Defender*. In contrast, the collecting of blocks works the same way in *Tower Of Power* and the *MoCo* games: physically moving over a block or a block spawning area.

Again, this brief breakdown of the results should be taken with caution as the number of participants was rather small. The complete data sheet with the answers from all players can be found on the enclosed CD-ROM/DVD (cf. Appendix B).

5.3 Expert Heuristic Evaluation

5.3.1 Method

The method chosen for the second evaluation is the *usability expert evaluation* (often also referred to as *expert heuristic evaluation*, *heuristic evaluation* or many other terms[1]).

Common user studies highlight issues specific to the game: boredom, challenge and pace level, as well as terminology. But usually these issues are targeted by usability engineers not until the more general problems in the interface and usability have already been addressed. Evaluations using heuristics (in contrast to user studies) do not target specific gameplay relevant elements such as specific behaviours and problems that could only be found by observing user play. Instead, heuristic evaluations target general system principles [10]. Therefore, it seems reasonable to perform an expert evaluation as a very first usability evaluation method.

It is important to understand that using heuristics is a discount usability method and it will not find all usability problems, but heuristics help to broaden the scope of an expert evaluation. Heuristics also give some credibility to the results evaluators find. Still, an expert evaluation does not automatically target all problems real users would have, so it is recommended to conduct user tests in addition. After all, usability engineering is an iterative process which will find more and more detailed issues over time. There is even a common rule which says *test early, test often.* This is especially true for heuristic evaluations [34].

The traditional procedure of an expert evaluation is divided into four phases [22, 28]:

1. **Kick-off meeting:** In the initial meeting the game is briefly introduced to the evaluators. The experts need to know who the target audience of the game is. They also need to know about missing features. As often video games are too big and content-rich to evaluate everything in a single session, another part of the meeting can be to agree on the issues that should be evaluated.
2. **Reviewing the game:** The evaluators use the provided set of heuristic rules, knowledge of good design practices and work related experience to review the application. In this step the specialists work independently as far as possible in order to find a broad range of issues.
3. **Evaluators review session:** The experts discuss their findings and compile a list of found issues and possible solutions. The goal is to clarify or eliminate potential false findings (wrong assumptions) and agree about severity of the remaining issues.

[1] http://www.uxmatters.com/mt/archives/2014/06/an-overview-of-expert-heuristic-evaluations.php

4. **Final report:** The final result of the expert evaluation is a categorized or quantified list of violations of the heuristics which helps the prioritizing during the problem solving.

Also for the conducted expert evaluation in the course of this thesis the final product is an evaluation report. For each found issue it lists a description, the violated heuristic, the frequency of occurrence, a severity rating and an effort rating including a suggested solution to the problem.

Heuristics

For creating an appropriate set of heuristics, Schaffer [34] recommends to start with one or more existing lists of heuristics and alter or improve their wording until they match given criteria. Therefore, a new set containing eight heuristics to evaluate usability and two heuristics to evaluate game play was assembled. The sources for the set were the usability and gameplay heuristics by Laitinen [22], the heuristics to evaluate playability (HEP) by Desurvire [10], the playability heuristics for mobile games by Korhonen [21] as well as the heuristics for usability and game mechanics by Pinelle [31]. As these heuristics overlap here and there, their meaning and description were unified in the respective cases.

The final set of heuristics used in the expert evaluation can be found in Appendix A.1. When compared to Jacob Nielsen's original heuristics for usability evaluations [28] one can see that despite their strong adaptation to video games the heuristics mirror the basic principles of Nielsen's work.

Experts, Planning and Preparation

Nielsen recommends to involve 3-5 experts in the evaluation [28]. Other researchers, however, such as Korhonen [20] and Laitinen [22] who are specialised in heuristic evaluations for video games have settled with 2-3 evaluators and also gained satisfactory results. In general it can be said that the number of found problems does not rise considerably when more specialists are involved, especially when going beyond 10 experts (cf. fig. 5.4). Also, the more evaluators participate in the evaluation the more expensive and challenging it will get to coordinate everything [28].

The people who conduct an expert evaluation are usually usability specialists and not necessarily experienced in game design. However, it does not hurt to have people in the team who are familiar with both fields. At least the person who leads the evaluation and works on the final report should, if possible, be a so called double expert (experienced in usability engineering and game design). Also, the evaluators should not be the same people who are working on the game as they know their work too well, and thus might be too forgiving or simply oversee specific issues which external specialists would find. [20, 22]

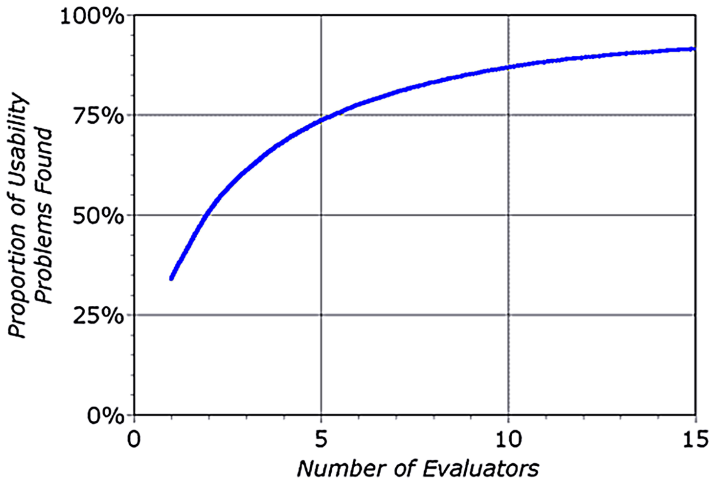


Figure 5.4: Proportion of number of evaluators to found usability issues [28, p. 156].

The actual evaluation took place in the *Ars Electronica Deep Space*: Two staff members from the *Ars Electronica Futurelab*, both with a professional background in multimedia and interactive installations, were personally instructed with the heuristics (cf. Appendix A.1) in order to conduct the evaluation. Additionally, they also received a handout with the heuristics as well as the severity and frequency metrics (cf. Section 5.3.1). They were given 20 minutes to play first *Lazor Arena* and then *Lazor Lab*, 10 minutes for each game. The devices they used were both *iOS* smartphones[2]. After playing the games they were asked to write down their findings and to rate the severity and frequency by using the predefined metrics from the handout.

Report Structure and Processing of Found Data

Usually, the final report is written by a single person, ideally one of the experts of the evaluation. As it can take a while (often a few days) to put all findings in the report, it is not required to involve more than one expert in this final process of documentation.

The complete report of the conducted evaluation can be found on the enclosed CD-ROM/DVD (cf. Appendix B.1). The results taken from the report can also be found directly in Appendix A.2. The report contains a list of the found issues. Each issue has a description which explains why it is problematic and in which situations it occurs, the id of the violated heuristic, a proposed solution and several additional parameters, listed below:

[2] *Apple® iPhone® 4*

Severity refers to the degree of obstruction in the usability. The severity rating has five grading steps[3]:

0 – No problem: A heuristic is violated but there are no consequences.

1 – Minor problem: The negative effect is negligible or purely aesthetic.

2 – Problem: The issue has a small impact on usability or playability.

3 – Major problem: The issue has a strong impact on usability or playability.

4 – Critical problem: A very serious problem, proper usage of the application is not possible. Also known as *show-stopper*.

Frequency of occurrence describes how often a user is affected by the issue. There are three possible grading steps:

1 – Rare: The issue only occurs in rare situations or only affects a small percentage of users.

2 – Often: The issue occurs in many situations or affects a large percentage of users.

3 – Permanent: The issue occurs all the time or affects every single user.

Ease of solution is the required effort to fix the issue. It has also three possible grading steps:

1 – Difficult: Profound changes have to be made in order to fix the issue. Other subsystems might be affected by the changes so that big chunks of the system need to be re-designed or re-implemented.

2 – Average: A small part of the system needs to be re-designed or re-implemented.

3 – Easy: Tiny changes to one or a few specific elements are sufficient in order to fix the issue.

Overall score: Finally, there is the overall score which helps to prioritize found issues. It is calculated from the above metrics. Given the *overall score* as o, *severity* as s, *frequency of occurrence* as f and *ease of solution* as e, the formula is: $o = s \cdot (f + e)$.

Unfortunately, this formula is not a commonly used approach to measure usability problems. It has not been used a lot in scientific studies so far and therefore its value prove is yet pending. A critical point is e.g. that the evaluators have to estimate the *frequency of occurrence* as there is no real user data available. Also the *severity* rating is rather objective, however, this is a common issue which all problem severity rating methods share. Ultimately, the approach was used anyhow as it was found to deliver results

[3]There are, however, many other ways to measure severity with various grading steps: http://www.measuringu.com/blog/rating-severity.php

which are easy to understand, easy to produce and of sufficient quality for the evaluation in the scope of this investigation.

Report Usage

The results in the report are sorted by the *overall score* and thereby already suggest an order in which the problems can be addressed. Issues with a high score should be fixed early as they have a high impact on the usability, occur often and are easy to solve. In the end, as already discussed, this sorting is not the ultimate truth but another important tool to help understanding the weight of the found problems.

Another way to approach the results in the report is to prioritize issues with a high rating in *ease of solution*. Focusing on the so called *low-hanging fruits*[4] is a great way improve usability quickly with little effort. However, care should be taken because sometimes focusing too much on *low-hanging fruits* may lead to turning a blind eye to the more fundamental problems which, again, might be the original cause of some of the smaller issues. Thus, getting to the root of the problem is often the better approach.

Often, after the report has been completed, a review session is organized in which one of the experts goes through the report together with the developers and talk about the findings. The expertise of both parties is often beneficial and helps to clear up misunderstandings. Ultimately, even if not all found issues are addressed, which is absolutely legitimate, the data helps to focus the development process and define goals for current iteration cycle [22]. In the course of the conducted evaluation, however, it was not necessary to hold an additional review session.

5.3.2 Results

Altogether, there were 13 issues reported by the experts. After merging the duplicates there were 12 issues left. During the debriefing the evaluators rated the severity and the frequency for each issue. In presence of one of the developers of *MoCo*, a proposed solution in combination with a rating for the ease of solution was added. Finally, everything was put into a report structure and the overall score was calculated. Interestingly, the range of severity ratings of all found problems only goes from 1 to 3, which means that no purely aesthetic issues (severity: 0) or critical problems (severity: 4) were found. There is not an extremely large number of violations of a specific heuristic or in a certain area, but rather the opposite is true: the found problems and their rating are harmoniously distributed. However, there were no violations found for the heuristics no. 3 (“The player’s memory load is minimized.”) and no. 7 (“Drop in and drop out works quickly and effortless.”).

[4] Commonly known as goals which are easy to achieve.

There are four problems with an overall score of 12, followed by three problems with a score of 10 and five more issues with a score under 10. In the following the problems are ordered by their overall score:

1. **(overall score: 12) Calibration point representation:** The first problem states that the meaning of the visual representation of the calibration point on the playing field is not clear. As a result, players might not understand how to calibrate their smart devices. Even though the calibration view (fig. 4.4 (c)) on the private screen contains a very similar icon (arrow pointing to the center of the calibration point), the geometric shape seems not to be clear enough. The obvious solution to this issue is to go away from the circle-shape and change the icon to a reduced and clearer x-shape.
2. **(overall score: 12) Sentry gun usability:** The next problem which was found states that using the sentry laser in *Lazor Lab* sometimes is abruptly aborted when the player accidentally leaves the trigger area of the sentry platform. This problem does not happen very often but it is still really frustrating for the player as he or she wastes three energy blocks all at once. The problem is also easy to fix by increasing the trigger size of the sentry platform.
3. **(overall score: 12) Tracking system delay:** A problem which is rather hard to fix is the delay of the tracking system. Heuristic no. 6 clearly states that controls should respond quickly. One of the experts noted that the delay of the tracking system makes it difficult to predict the players' positions and especially in *Lazor Arena* the game becomes about anticipating the slowly moving player avatars instead of strategic and responsive movement. As the delay is caused on the server side of the tracking implementation, it is not possible to approach this problem through *MoCo*. A workaround could be to alter the game design in order to decrease the game pace so that the latency does not affect the game significantly.
4. **(overall score: 12) Health core state:** Another problem is that the state of the health core in *Lazor Lab* is only visualized through color. The color fades from green (healthy) to red (destroyed) the more damage it receives. As it's hard to tell the actual health during the game, tension and urgency gets lost. This problem affects every player all the time but it is actually easy to fix: The proposed solution is to simply add a progress bar or a numerical value to the GUI on the public display.
5. **(overall score: 10) Range of laser beams:** When releasing a grabbed energy block a laser beam is fired. The range of this beam is not communicated clearly enough. This leads to situations in which players assume they effectively hit another player in *Lazor Arena*, but instead it is a miss. In combination with the delayed movement of avatars, due

to latency, this issue causes frustration. To solve this problem, either the range of the laser beams needs to be increased or there could be some kind of crosshair for each player.

6. **(overall score: 10) Repairing wall usability:** In *Lazor Lab* the color a brick is expected to get repaired with is not visualized. Also, there are no cues whatsoever indicating that two bricks of the same color will explode when placed directly next to each other. This lack of transparency naturally causes frustration, as the player is not prevented from making mistakes. To approach this issue, either some kind of tutorial could be introduced or a visualization of the color of the broken wall slot could be shown.
7. **(overall score: 10) Enemy representation:** In *Lazor Lab* the visual representation of the enemies is problematic. As the hostile polygons are grey colored they do not stand out from the white background very good. This makes them hard to see or recognize. Changing the visual appearance (color, size or shape) or adding some kind of eye-catching details (blinking highlights or a trail) could solve this issue.
8. **(overall score: 9) Player state management:** Sometimes it happens that the laser ranger system loses track of a player. This usually only occurs when a player moves close to the edges of the playing field, runs fast or jumps. However, the two demo games do not cover these situations sufficient. When a player loses his or her tracking and then re-identifies him or herself, in *Lazor Arena* the player score is restored, but in both games the player loses all collected energy blocks. To solve this problem the game could save the players' game states, for identifying the correct game state on reconnection either the player name or the hardware id of the smart device could be used. However, a better approach would be to extend the underlying tracking system to wait for the tracking record to appear again near by before the tracking record gets removed entirely together with the player avatar.
9. **(overall score: 6) Complex body coordination:** A rather peculiar problem is that some players experience troubles coordinating head-hand orientation and body orientation while walking/running. This problem is hard to solve as it is a core mechanic of *Lazor Arena* and *Lazor Lab* to use the mobile device as a pointer. The only ways to handle this issue are to replace the pointer-mechanic and utilize the gyroscope sensor for something less elementary or to completely remove the gyroscope controls. The reason why this issue has a relatively low overall score even though it seems rather severe is the bad rating in ease of solution.
10. **(overall score: 5) Communication of possible actions:** In *Lazor Lab* some of the possible player actions are not communicated properly. For example, the possibility to repair a wall or to use a sentry

gun is not being visualized on the public display. This leads to a lack of affordance during the game and players might be confused about what actions they can or should perform. Adding visual cues directly at the particular spots were user actions are required would solve this problem.

11. **(overall score: 4) Inconsistent collectables:** The way collectables (energy blocks) spawn in the two demo games is different. In *Lazor Arena* the blocks spawn randomly anywhere on the playing field, in *Lazor Lab* there are five specific spawn areas which randomly produce energy blocks. This lack of consistency is problematic because players might get wrong expectations of what the collectables do and how they work. This issue can be fixed by unifying the way energy blocks spawn.
12. **(overall score: 3) Player-to-player communication:** The issue with the lowest overall score states that the communication with other players is difficult. Especially in *Lazor Lab* in which players have to collaborate, manage resources and distribute tasks in order to survive, it is difficult to tell what tasks other players are currently following. There are two ways to approach this issue: When reducing the game pace by altering the game design, there should be more time to actually talk with each other while playing. The other way is to introduce some kind of visual and/or acoustic commands which can be triggered through the smart devices (e.g. “Defend the core!” or “I need a blue block!”).

As expected, the expert evaluation shed light on some usability problems of *MoCo* and playability problems of the two demo games. Some more discussion on the results of both evaluations can be found in the final Chapter 6. However, in order to cover a more complete spectrum of usability and playability issues, larger user play tests as well as further heuristic evaluations throughout the whole development process of *MoCo* are required.

Chapter 6

Concluding Debate

6.1 Results Analysis

Both evaluations, the game experience survey and the expert heuristic evaluation, provide valuable data to improve *MoCo* and its games. In the following are some interesting findings and conclusions:

Good overall results: In general, it can be said that the evaluations received good results and the system was relatively well received by the users who filled out the survey but also by the experts who conducted the heuristic evaluation. In the survey all questions received an arithmetic mean greater than 3, which means that for every question the average user either agrees or does neither agree or disagree. In the expert evaluations no critical problems were found and only three issues were rated as a major problem.

Bad audio balancing in test setting: One thing which probably could have been avoided is the rather bad result for Q5 in the game experience survey: As already mentioned the balancing of the sound levels in the test environment was anything but good. The audio output from the game on the public display which came from the room-installed sound system in the *Deep Space* was quite loud compared to the small speakers of the used smart devices (*iPhone 4* smartphones). A proper balancing of the audio levels would have lead to more players recognizing the distinction between public audio feedback and individual private audio feedback.

Few system related problems: An essential finding is that only 2-3 of the problems (depending on how they are interpreted) are actually related to *MoCo* or the underlying system: Merely the unclear visual representation of the calibration point, the current handling of players who get lost by the tracking system (which actually is not really part of *MoCo*) and the delay of the tracking system (again not part of *MoCo*) are issues on the system level. The rest of the issues is purely game

related.

Identification process proven to work: Another vital finding is, that one of the most important features of *MoCo* was proven to work flawlessly: the identification process. Both the players in the test session as well as the experts in the heuristic evaluations immediately understood how to identify their tracking record with their smart device. Even with up to six players simultaneously, if a player got lost by the tracking server he or she was able to quickly re-identify him or herself.

Differences in accessibility: A quite interesting impression was gathered when people were observed playing *Lazor Arena*, *Lazor Lab* and *Game Changer*: The games based on *MoCo* seem to vary in terms of accessibility from games with merely body controls, presumably because of the additionally required smart device. On the one hand, people tend to enter the playing field automatically on their own more likely if there are no additional devices required to start playing, on the other hand, however, people who seem interested in the game but are yet reserved are more easily to persuade to try playing when they are offered a device directly through a supervisor on-site.

Client UI has room for improvement: Another observation was made during the user play tests: Once the players understood the interface on the private display, they could operate it more or less blindly. The generously scaled buttons are easy to press without looking on the smart device. However, many players needed support from one of the supervisors as they did not understand that some buttons are meant to be pressed and held (e.g. for grabbing and carrying an energy block). Either some kind of tutorial for first-time users or better labelling of the UI elements could solve this issue.

Some of these findings are not direct results from the evaluations but rather have been made through observation and inference. This clearly shows that there are many more valuable data to collect from co-located smart games and a lot of research still has to be done to scientifically substantiate such assumptions.

6.2 Further Prospects

MoCo obviously has some flaws which are clearly pointed out by both evaluations. However, these issues are solvable. For some of the problems the solutions are even documented in the Chapters 2 and 3 already. Unfortunately, due to a lack of time and the fact that *MoCo* is more of a prototypic proof of concept than a final product, many suggested features did not find their way into the system yet. However, they are already on the list for further development of *MoCo* (or its successor):

- More visual or acoustic cues to steer the users' attention between the displays.
- A comprehensive personal user profile synchronized between server and client.
- A sophisticated architecture for keeping track of the player states (especially for re-establishing the progress after disconnections).
- Exposing the *MoCo* server settings (port, max clients and NAT Punch-through usage) to the XML configuration file.
- More diverse input modalities (e.g. sensors such as camera, microphone, accelerometer or even UI elements such as sliders or input fields).
- Porting the client to more mobile platforms such as *Windows 10 Mobile*.
- Decoupling the server and client code even more in order to foster a clean system architecture and to improve the ability to reuse and extend the framework.

But not just *MoCo* has room for improvements, also the *Unity TUIO/Pharus Tracking Client* (cf. Section 4.2.1) has new features already in the pipeline: The most important addition is to not trigger the `OnTrackLost` event (cf. Fig. 4.2) immediately when a track gets lost, but instead to introduce a new idle/pending event. Pending entities are basically lost, but their player object will remain invisible until it either gets eventually removed after a while of inactivity (`OnTrackLost` event) or a new tracking record appears near by and the pending record will replace the new one and thereby the invisible player object will become active again. This feature should prevent players from losing their game progress due to the tracking system through a low level implementation instead of an extra specific implementation for every game.

6.3 Conclusion

The prior work in the field of co-located floor-based video games as well as the research in smart gaming and co-located play were both essential parts when designing novel interaction modalities: The creation of *MoCo*, a system which extends the actual player interface to the game, would not have been possible without the appropriate effort. However, by allowing players to perform more diverse actions, *MoCo* is only one way to give more control to the players. Augmented reality (AR) and virtual reality (VR) are on their way and without any doubt they will turn video games and very likely also co-located games upside down. Both technologies offer exciting novel mechanics to alter or enhance the user experience drastically. And of

course there are still other systems which extend their user interface through proprietary additional hardware, such as the *Wii BalanceBoard*[1], the *Wii Remote*[2] [19] or the *Kinect*[3]. However, in some cases the hardware is still too rigid or does not properly scale for large dimensioned environments such as the *Deep Space*. Also, in the author's opinion smart devices still offer the best results in terms of accessibility, as nearly everyone these days constantly carries at least one smart device. Furthermore, smart devices are becoming more and more sophisticated and feature-rich and as for now there is no end in sight for this trend.

In any case, one thing seems certain: There are going to be some exciting combinations of various new technologies which will pave the way for a bright future for co-located video games and the author hopes that this thesis takes us one step further on this long road ahead.

[1] *Nintendo® Wii™ BalanceBoard*
[2] *Nintendo® Wii™ Remote*, often referred to as *Wiimote*
[3] *Microsoft® Kinect®*

Appendix A

Results: Expert Heuristic Evaluation

A.1 Used Heuristics

Table A.1 shows the assembled set of heuristics used in the expert heuristic evaluation. It is referenced in the Sections 5.3.1 and 5.3.1.

A.2 Evaluation Results

Table A.2 holds the results of the expert heuristic evaluation. It is referenced in Section 5.3.1. The results are taken directly from the evaluation report which can be found on the enclosed CD-ROM/DVD (cf. Appendix B.1).

1	**Appropriate feedback is provided.**	The game provides immediate, adequate, and easy-to-understand feedback after each action taken within the game or while using menus before or after playing the game. The provided feedback supports understanding the consequences of the action. (Such actions are e.g. a single press of a button, complicated input sequences like combos or the character interacting with the environment/gameworld)
2	**Terminology and visual language are easy to understand.**	The terminology and language used in the game and in menus is easy to understand. Technical terms are avoided. Texts are written from the player's point of view. Art, symbols and pictograms in the interface are either commonly known or easy to recognize (and speak to its function).
3	**The player's memory load is minimized.**	The player is not required to remember information. The information the player needs is displayed clearly at the time the player needs it. User preferences and achievements are saved across game sessions. Frequently required information is placed prominently in the interface or effortless accessible.
4	**The player is prevented from making mistakes.**	The user interface is designed in a way so that it prevents the player from making mistakes that are not part of the gameplay (especially irreversible errors). The user interface limits available options, provides help or automates options in order to reduce the number of errors the players can make. If errors occur, easy-to-understand error messages (that informs the player about the consequences of the error and what the player can do to recover from the error) are provided.
5	**The audiovisual representation supports the game.**	Graphics and visual effects are used to make the basic user interface elements easy to use and appealing. The audiovisual appearance and tactile effects of the game supports also the gameplay by providing information and feedback.
6	**The game controls are convenient and respond quickly.**	Controls are easy to learn and accessible for people who do not play games often. They are suitable for the game and offer efficient ways to take actions in the game. Input mapping is designed in a way that users can issue commands quickly and accurately in order to respond rapidly and intuitively to game events. Shortcuts are provided for common activities (if applicable).
7	**Drop in and drop out works quickly and effortless.**	Joining and leaving the game works easily and quickly. Dropping in or out during an active game session does not affect the experience for other players.
8	**Interruptions are handled reasonably.**	The game handles interruptions properly. In case of an interruption or disconnection the re-establishment of the previous state is handled fast and as user-friendly as possible.
9	**The player is in control.**	The game conveys the feeling that the player is in control. Unforeseeable events with negative effects on the player's progress are reduced to a minimum. The player never starts feeling that the game is more of a lottery than a game.
10	**Progress and results are visible during the game.**	The player is always informed on his or her (or the team's) progress within the game. By being able to compare results and progress motivation is maintained.

Table A.1: The final set of heuristics used by the experts to evaluate *MoCo* and its demo games.

Description	Heur.	Sev.	Freq.	Proposed Solution	Ease of solution	Overall score
The meaning of the visual representation of the calibration point on the playing field is not clear.	2	2	3	Change the visual representation of the calibration point to something more obvious.	3	**12**
When using the sentry laser in Lazor Lab it sometimes happens that the actions is aborted because the player accidentally left the trigger area.	4	3	1	Either increase the size of the trigger area or allow shooting the big laser outside of the trigger area once it was started.	3	**12**
Delay of the tracking system makes is hard to predict the players' positions. In Lazor Arena the game becomes more about the slowly moving player circles instead of the actual people.	6	3	3	Either reduce latency, or decrease game pace so that the delay does not play such a significant role..	1	**12**
The state of the core in Lazor Lab is only visualized by color. This is not sufficient to tell the current "game health". This leads to a slight lack of tension / urgency.	10	2	3	Add a progressbar or percentage to the health core so that the game state is clearly visible.	3	**12**
The range of the laser beam is not communicated clearly enough. This leads to frustration, when an assumed hit does not actually hit.	1	2	2	Either increase range of lasers or add some kind of visual crosshair.	3	**10**
In Lazor Lab the color a wall brick is expected to get repaired with is not visualized. This leads to players building bricks with wrong colors and causes frustration when the bricks explode as a consequence.	4	2	3	Either visualize the color of a broken wall slot or explain the repair mechanic better through a tutorial.	2	**10**
Color and shape of hostile polygons in Lazor Lab makes them hard to see/recognize.	5	2	3	Change the visual appearance of enemies, add some eye-catching details. (loud colors, blinking, trail)	2	**10**
Sometimes it happens that the tracking system loses a player, when this happens the player can easily reconnect but loses his or her collectables.	8	3	1	Save the players' game state on the server side. Once a player reconnects reestablish his or her progress.	2	**9**
It is often difficult to coordinate head-hand orientation and body orientation while walking/running.	6	2	2	Remove or replace pointer-mechanic (gyroscope controls) entirely.	1	**6**
Some player actions are missing visual affordance on the public display. E.g. repairing a wall / using sentry gun in Lazor Lab.	1	1	2	Add visual cues on the public display to increase affordance.	3	**5**
The way the spawning of collectables (energy blocks) in the two demo games work differently. This leads to different expectations although the actually work the same way.	2	1	2	Uniform the way collectables appear in the game world.	2	**4**
Communication with other players in Lazor Lab often is difficult as it's hard to tell what tasks other players are currently follwing.	9	1	2	Introduce some kind of visual or acoustic commands which can be triggered through the smart devices (e.g. "Defend the core!", "I need a blue block!").	1	**3**

Table A.2: The results list taken from the evaluation report. The found problems are ranked after the overall score (cf. Section 5.3.1), and thus already suggest an order to approach the issues.

Appendix B

Content of the CD-ROM/DVD

Format: CD-ROM, Single Layer, ISO9660-Format

B.1 Evaluations

Pfad: /

Friedl_Andreas_2015.pdf Thesis (this document)

Pfad: /GameExperienceSurvey

Handout.pdf	The actual survey which was handed out to the play testers
Results.pdf	The complete results of the game experience survey

Pfad: /UsabilityExpertEvaluation

Handout.pdf	The handout the experts received for the evaluation
Heuristics.pdf	A list of all heuristics used in the usability expert evaluation
Report.pdf	The final evaluation report including raw findings as well as the final results list
Results.pdf	The raw results of the evaluation in a plain table

B.2 Project

Pfad: /Project/Bin

LazorLab_Client_(Android).zip MoCo client executable (Android)
LazorLab_Server_(Win).zip MoCo server executable (Windows)

Pfad: /Project/Other

MoCo_early_prototype.mp4 Video of an early prototype of MoCo
GameDesign_LazorLab.pdf Early design draft for LazorLab

Pfad: /Project/Src

UnityProject_Client_(Android_iOS).zip Project files for the MoCo client (Android and iOS)
UnityProject_Server_(Win).zip Project files for the MoCo server (Windows)
UnityProject_TrackingServer_(Win).zip Project files for the plain UnityTrackingServer framework (Windows)

B.3 Miscellaneous

Pfad: /Images

*.jpg, *.png . . . . . original raster images
*.pdf . . . . . original vector images and tables

References

Literature

[1] Rafael Ballagas et al. “BYOD : Bring Your Own Device”. In: *Proceedings of the Workshop on Ubiquitous Display Environments, UbiComp '04*. Nottingham, England, UK, 2004, pp. 20–28 (cit. on pp. 5, 15, 28).

[2] Rafael Ballagas et al. “The smart phone: A ubiquitous input device”. In: *IEEE Pervasive Computing* 5.January (2006), pp. 70–77 (cit. on pp. 19, 23).

[3] Victoria Bellotti and Keith Edwards. “Intelligibility and Accountability: Human Considerations in Context-Aware Systems”. In: *Human-Computer Interaction* 16.2 (2001), pp. 193–212 (cit. on p. 24).

[4] Harry Brignull and Yvonne Rogers. “Enticing People to Interact with Large Public Displays in Public Spaces”. In: *Proceedings of International Conference on Human-Computer Interaction*. Zürich, Switzerland, 2003, pp. 17–24 (cit. on pp. 20, 22, 23).

[5] Darryl Charles et al. “Player-centred game design: Player modelling and adaptive digital games”. In: *Proceedings of DiGRA 2005 Conference: Changing Views – Worlds in Play*. Vancouver, BC, Canada, 2005, pp. 285–298 (cit. on p. 29).

[6] Victor Cheung et al. “Overcoming Interaction Barriers in Large Public Displays Using Personal Devices”. In: *Proceedings of the 9th ACM International Conference on Interactive Tabletops and Surfaces, ITS '14*. Dresden, Germany: ACM Press, 2014, pp. 375–380 (cit. on pp. 20–25, 27, 29).

[7] Herbert H Clark and Susan E Brennan. “Grounding In Communication”. In: *Perspectives on Socially Shared Cognition*. Vol. 13. Washington, DC, USA: American Psychological Association, 1991. Chap. 3, pp. 127–149 (cit. on p. 4).

[8] Nigel Davies et al. “Using Bluetooth Device Names to Support Interaction in Smart Environments”. In: *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*. Wroclaw, Poland: ACM Press, 2009, pp. 151–164 (cit. on p. 23).

[9] David Dearman and Khai N Truong. “BlueTone: A Framework for Interacting with Public Displays Using Dual-tone Multi-frequency Through Bluetooth”. In: *Proceedings of the 11th International Conference on Ubiquitous Computing*. Orlando, FL, USA: ACM Press, 2009, pp. 97–100 (cit. on p. 23).

[10] Heather Desurvire, Martin Caplan, and Jozsef A. Toth. “Using Heuristics to Evaluate the Playability of Games”. In: *Proceedings of the Conference on Human Factors in Computing Systems, CHI ’04*. Vienna, Austria: ACM Press, 2004, pp. 1509–1512 (cit. on pp. 53, 54).

[11] Jeremiah Diephuis et al. “Game Changer: Designing Co-Located Games that Utilize Player Proximity”. In: *Proceedings of DiGRA 2015: Diversity of Play*. Lüneburg, Germany, 2015 (cit. on pp. 1, 26).

[12] Katharina Emmerich, Stefan Liszio, and Maic Masuch. “Defining Second Screen Gaming: Exploration of New Design Patterns”. In: *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*. Funchal, Portugal: ACM Press, 2014 (cit. on pp. 19, 21, 25–29).

[13] Tracy Fullerton and Christopher Swain. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. 2nd ed. Taylor & Francis, 2008 (cit. on pp. 27, 29).

[14] Saul Greenberg et al. “Dark Patterns in Proxemic Interactions: A Critical Perspective”. In: *Proceedings of the 2014 Conference on Designing Interactive Systems*. Vancouver, BC, Canada: ACM Press, 2014, pp. 523–532 (cit. on p. 24).

[15] Kaj Grønbæk et al. “IGameFloor: A Platform for Co-located Collaborative Games”. In: *Proceedings of the International Conference on Advances in Computer Entertainment Technology*. Salzburg, Austria: ACM Press, 2007, pp. 64–71 (cit. on pp. 5, 9, 10).

[16] Wolfgang Hochleitner, Michael Lankes, and Christina Hochleitner. “Limelight – Fostering Sociability in a Co-located Game”. In: *Proceedings of the Workshop on Designing and Evaluating Sociability in Online Video Games, CHI ’13*. Paris, France: ACM Press, 2013, pp. 23–28 (cit. on pp. 4, 12, 17, 28).

[17] Ryo Izuta et al. “Early Gesture Recognition Method with an Accelerometer”. In: *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia, MoMM ’14*. Kaohsiung, Taiwan: ACM Press, 2014, pp. 43–51 (cit. on p. 9).

[18] Martin Kaltenbrunner et al. “TUIO - A Protocol for Table-Top Tangible User Interfaces”. In: *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation*. Vannes, France: Springer, 2005 (cit. on pp. 10, 37).

[19] Dennis L Kappen et al. “Exploring Social Interaction in Co-Located Multiplayer Games”. In: *Proceedings of the Conference on Human Factors in Computing Systems, CHI ’13*. Paris, France: ACM Press, 2013, pp. 1119–1124 (cit. on p. 64).

[20] Hannu Korhonen. “Comparison of Playtesting and Expert Review Methods in Mobile Game Evaluation”. In: *Proceedings of the 3rd International Conference on Fun and Games*. Leuven, Belgium: ACM Press, 2010, pp. 18–27 (cit. on p. 54).

[21] Hannu Korhonen and Elina Koivisto. “Playability Heuristics for Mobile Games”. In: *Proceedings of the 8th Conference on Human-computer Interaction with Mobile Devices and Services, MobileHCI ’06*. Espoo, Finland: ACM Press, 2006, pp. 9–16 (cit. on p. 54).

[22] Sauli Laitinen. “Usability and Playability Expert Evaluation”. In: *Game Usability: Advancing the Player Experience*. Ed. by Morgan Kaufmann. CRC Press, 2008. Chap. 7, pp. 91–111 (cit. on pp. 53, 54, 57).

[23] Pat Langley. “Machine Learning for Adaptive User Interfaces”. In: *Proceedings of the 21st German Annual Conference on Artificial Intelligence*. Freiburg, Germany: Springer, 1997, pp. 53–62 (cit. on p. 29).

[24] Jaana Leikas, Antti Väätänen, and Veli-pekka Räty. “Virtual Space Computer Games with a Floor Sensor Control Human Centred Approach in the Design Process”. In: *Proceedings of the 1st International Workshop on Haptic Human-Computer Interaction*. Glasgow, Scotland, UK: Springer, 2000, pp. 199–204 (cit. on pp. 5, 8).

[25] Jaana Leikas et al. “Multi-User Mobile Applications and a Public Display: Novel Ways for Social Interaction”. In: *Proceedings of the 4th Annual IEEE International Conference on Pervasive Computing and Communications*. Pisa, Italy: IEEE, 2006, pp. 66–70 (cit. on pp. 20, 24, 29).

[26] Jörg Müller et al. “Display Blindness: The Effect of Expectations on Attention towards Digital Signage”. In: *Proceedings of the 7th International Conference on Pervasive Computing, Pervasive ’09*. Nara, Japan, 2009 (cit. on p. 22).

[27] Otto Naderer. “Crowd Tracking And Movement Pattern Recognition”. master thesis. Johannes Kepler University Linz, 2015. URL: http://epub.jku.at/obvulihs/content/titleinfo/493866 (cit. on pp. 5, 10, 16, 36, 37).

[28] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann, 1993 (cit. on pp. 53–55).

[29] Timo Ojala et al. “Multipurpose Interactive Public Displays in the Wild: Three Years Later”. In: *Computer* 45.5 (2012), pp. 42–49 (cit. on p. 20).

[30] Susan Palmiter, Jay Elkerton, and Patricia Baggett. “Animated Demonstrations vs Written Instructions for Learning Procedural Tasks: A Preliminary Investigation”. In: *International Journal of Man-Machine Studies* 34.5 (1991), pp. 687–701 (cit. on p. 22).

[31] David Pinelle, Nelson Wong, and Tadeusz Stach. “Heuristic Evaluation for Games: Usability Principles for Video Game Design”. In: *Proceedings of the Conference on Human Factors in Computing Systems, CHI ’08*. Florence, Italy: ACM Press, 2008, pp. 1453–1462 (cit. on p. 54).

[32] Stefan Poslad. *Ubiquitous Computing: Smart Devices, Environments and Interactions*. Wiley, 2009 (cit. on p. 19).

[33] Stuart Reeves et al. “Designing the Spectator Experience”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’05*. Portland, OR, USA, 2005, pp. 741–750 (cit. on pp. 20, 23).

[34] Noah Schaffer. “Heuristic Evaluation of Games”. In: *Game Usability: Advancing the Player Experience*. Ed. by Morgan Kaufmann. CRC Press, 2008. Chap. 6, pp. 79–89 (cit. on pp. 53, 54).

[35] Stacey D Scott and James R Wallace. “"Local Remote" Collaboration: Applying Remote Group AwarenessTechniques to Co-located Settings”. In: *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing*. Vancouver, BC, Canada: ACM Press, 2015, pp. 319–324 (cit. on p. 4).

[36] George Simmel. “The Sociology of Sociability”. In: *The American journal of sociology* 55.3 (1949), pp. 254–261 (cit. on p. 4).

[37] Scott S. Snibbe and Hayes S. Raffle. “Social Immersive Media: Pursuing Best Practices for Multi-user Interactive Camera/Projector Exhibits”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’09*. Boston, MA, USA: ACM Press, 2009, pp. 1447–1456 (cit. on p. 4).

[38] Hanna Strömberg, Antti Väätänen, and Veli-pekka Räty. “A Group Game Played in Interactive Virtual Space: Design and Evaluation”. In: *Proceedings of the 4th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*. London, England, UK: ACM Press, 2002, pp. 56–63 (cit. on pp. 5, 8).

[39] John Sweller, Jeroen van Merrienboer, and Fred Paas. “Cognitive Architecture and Instructional Design.” In: *Educational Psychology Review* 10.3 (1998), pp. 251–296 (cit. on p. 26).

[40] Daniel Vogel and Ravin Balakrishnan. “Interactive Public Ambient Displays: Transitioning from Implicit to Explicit, Public to Personal, Interaction with Multiple Users”. In: *Proceedings of the 17th annual ACM symposium on User interface software and technology, UIST '04*. Santa Fe, NM, USA, 2004, pp. 137–146 (cit. on p. 22).

[41] Huijing Zhao and Ryosuke Shibasaki. “A Novel System for Tracking Pedestrians using Multiple Single-Row Laser-Range Scanners”. In: *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* 35.2 (2005), pp. 283–291 (cit. on p. 10).