Actuated Ball Sports

Ben Clerix

Advisor: Prof. dr. Kris Luyten
Supervisor: Raf Ramakers

Thesis proposed to achieve the degree of master
in computer science: Human-Computer Interaction

Academic year 2014-2015

Acknowledgments

Hereby, I would like to thank everyone who helped me to make this thesis.

First of all, I would like to thank Prof dr. Kris Luyten for supervising this thesis. He inspired me to choose this subject and gave me a lot of helpful tips.

Also, I would like to thank Raf Ramakers for all the feedback he gave me. I could always, at any time, ask for advice and support. He had an answer to all the problems I encountered. He guided me with a lot of enthusiasm through this journey.

I would like to thank Prof dr. Peter Quax for the technical support with the different hardware.

I would like to thank Oeve Geypen for designing the custom quadcopter and for sharing all his knowledge regarding quadcopters.

Finally, I would like to thank my parents for supporting and helping me, not only with the creation of this thesis, but through my whole life.

Abstract

Ball sports offer many physical, mental and social benefits, however they each require a different set of attributes, such as basketball hoops, volleyball nets, bowling pins, etc. Thus in order to play each of these ball sports, we need to provide and manually set up each of different attributes. Digital ball sports on the other hand virtualise these different attributes at the cost of losing physicality, however they have the possibility to create non-realistic experiences. Without physicality players cannot use the ball coordination skills they developed over their lifetime.

In this thesis, we combine the tangibility of traditional ball sports with the flexibility of virtual games. Our system allows us to render the different attributes for each ball sport virtual. This allows us to implement multiple games in the real world without the hassle of manually setting up attributes. In this system multiple games can co-exist in a single general-purpose ball sports platform. In order to preserve the tangibility of the ball sport, the ball remains physical. We want to preserve the possibility of using non-realistic experiences often used in digital ball sports such as changing the gravity or the trajectory of the ball. Therefore we use an actuated ball in order to introduce non-realistic experiences into the real world.

We succeeded in creating a general-purpose ball sports platform in which we can create non-realistic experiences although these are limited by the performance of the actuated ball. We believe, given the proper actuated ball, it is possible to fully realise the presented system.

Contents

Chapter 1

Introduction

Video games have changed a lot over the last few years. In the past, most games involved people sitting in front of a pc or tv screen interacting with remote controls and peripherals. The release of the Wii in 2006 created a whole new experience because it was the first commercial system that made players use their whole body when playing a game. The Kinect was introduced a few years later, which is able to track your movement without the need for a physical controller. In both systems the player still has to look at the screen for most feedback, therefore there is less social interaction with other players compared to real world sports or games. Some systems like the one from Benko et al. [4] remove the screen completely. Their system projects a virtual ball in the real world which the players interact with. Because the ball is projected in the real world, instead of being shown on the screen, the players observe and engage with each other instead of the screen. Therefore this system stimulates team play and is more socially engaging than Wii or Kinect games.

Playing ball sports in the real world often involves physical attributes, such as basketball hoops, volleyball nets and balls. While interacting with balls during sports, the human body develops several skills that improve coordination and control. This is exemplified by skilful movements that we see in many ball sports, such as juggling a ball on one foot or dribbling with the ball. Balls are used in many sports throughout history, they appear on age-old drawings [1] and are used in ancient chinese ball sports [1]. Even nowadays new ball sports are invented almost every day [2]. Since many digital sport games have no physical attributes they lack physicality, therefore people are not able to use the precise coordination and control that they developed with physical attributes over a lifetime. Examples include, the system from Benko et al. and games on the Nintendo Wii and Microsoft Kinect, such as Boom Ball [3], where users use mid-air gestures to control a ball instead of interacting with a real physical ball.

In contrast to the previous systems in which sports are introduced in games, exertion games include digital game elements in real world sports. Exertion games are attributed with many physical, mental and social benefits, thus changing the way we play computer games[2]. In most exertion games players need to interact with physical objects, therefore they develop various skills. An example of such a game is the system developed by Mueller et al. [9]. In this system the player plays table tennis against two other players by hitting targets a number of times. However, such an approach does not allow for a single general-purpose gaming platform that can be used for many ball sports.

Exertion games allow for physical attributes just like traditional ball sports. There are several ways of categorising traditional ball sports. Firstly classification by number of players in a team, such as single player ball sports or team ball sports. A second

[1] http://en.wikipedia.org/wiki/Cuju
[2] http://en.wikipedia.org/wiki/List_of_ball_games
[3] http://www.boomballgame.com/

classification is based on the type of the ball sport[4], we can select the following broad types: bat-and-ball sports, racquet and ball sports, goal sports, net sports. In the context of this thesis, we divided traditional ball sports in two categories based on the way of interacting with the ball. The first group are ball sports which rely solely on direct interaction with the ball, thus the only physical attribute the player interacts with is the ball itself. Secondly there are ball sports where the player needs an attribute to interact with the ball like in tennis, baseball or squash. In this thesis we focus on the first group, thus meaning direct interaction with the ball, to preserve the tangibility of the ball. A lot of these direct interaction ball sports require a different set of attributes, rules, terrain and a different ball. The primary ball sports attribute is the ball which is always present, however the characteristics of the ball differ between ball sports. For example the size, weight and texture of basketballs are different from those of bowling balls. The secondary ball sport attributes, such as basketball hoops, volleyball nets and bowling cones are specific to each ball sport. In order to play each of these ball sports these physical attributes are required.

Our system consists of a tracking system and an actuated robotic ball. The actuated robotic ball is a flying ball that we can control remotely. By controlling the ball we change how it behaves and engage in the gameplay. Since people have been playing ball sports with all kind of balls throughout history, we assume it does not matter what the ball is made off to assure a challenging and amusing experience. The tracking system provides sub millimetre absolute positioning and tracks the position of the ball and the players. The main reason we use the tracking system is to control the actuated ball and correct it's movements for this we need it's position and orientation. Our system is a general-purpose ball sports platform and therefore supports a wide variety of games including basketball, volleyball, korfball and bowling. To allow multiple games to be deployed in a similar room environment we make all secondary ball sport attributes, such as basketball hoops, volleyball nets and bowling cones virtual. These virtual objects are invisible in the real world or projected. The primary ball sport attribute, the actuated ball, remains physical to address the dexterity of the human body and the skills humans developed with these balls over a lifetime.

There are already a number of systems that use actuated balls in their concept, such as HoverBall [14]. HoverBall introduced the first concepts of controllable robotic balls and uses gestures to control the ball. Actuated balls are relevant because we introduce non-realistic experiences to a physical ball in the real world. For example make the ball evade or follow certain players.

The research challenge in this thesis is to combine the tangibility of traditional ball sports with the flexibility of virtual games so that multiple games can co-exist in a single game platform without the need to reconfigure the entire environment and move secondary sports attributes, such as basketball hoops or nets, around. Besides this, we bring non-realistic experiences such as changing the gravity or the trajectory of the ball, to our platform by using an actuated ball. Actuated ball games is a generalisation of exertion games in that it preserves the flexibility of traditional virtual games.

1.1 Goal of this thesis

This thesis is a feasibility study in which we explore the opportunities of a general-purpose ball sports platform. In this section we elaborate on the rationale behind our system and on the goals we need to achieve.

We create a general-purpose ball sports platform because we want to combine the advantages of digital ball sports with the advantages of their real world counterparts. Digital ball sports support a wide variety of games without the requirement of physical secondary sports attributes and without the hassle of manually positioning them. Digital

[4] http://en.wikipedia.org/wiki/Ball_game

sports games often include unrealistic experiences, such as unrealistic physics and power-ups and they are not limited due to realism requirements. Digital sports games are often less socially engaging compared to real world activities since players experience the world only through a screen. Since players mainly focus on the screen they are unable to see the facial reactions of the other players which often limits social interaction. With the intention to increase the social interaction some systems remove the screen so players can engage more with each other.

Feedback in real world sports is easy to achieve compared to digital sport games since players are part of the world instead of experiencing the world from outside through a screen. Real world sports are also very socially engaging because players can interact with each other, though they are susceptible to real world physics, which is not the case in digital sport games. Physical attributes have the advantage that players can develop certain skills while interacting with them or use various skills they built up over their lifetime, which improves coordination and control. Ideally our aim is to create a system that combines the best of both digital world and real world ball sports. Next we discuss the goals we need to achieve to realise this system.

The main goal of this thesis is to develop the code framework that allows players to play actuated ball games using the general-purpose ball sports platform. The system allows the players to play several ball sports with direct interaction without the need for secondary sports attributes, such as basketball hoops, volleyball nets and bowling cones. To achieve this goal we define the secondary sports attributes only in the virtual world, for example, invisible or projected. We allow players to define the position of the the secondary sports attributes themselves by holding a marker in the air.

The software engages in the gameplay in several ways by controlling the actuated ball. Examples include helping the player achieve a goal, rewarding or punishing players or balancing player skills. In the context of this thesis we focus on helping the player achieve a goal, in this case throwing a basketball into a basketball hoop.

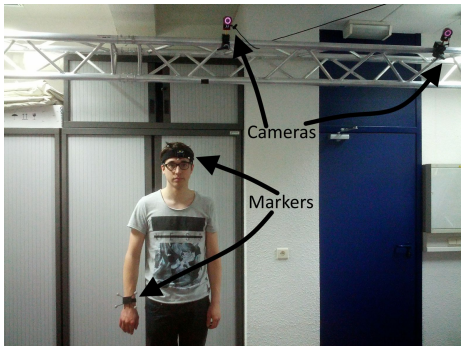
In our system we use an actuated ball, a tracking system and our software. In the end we want to accomplish a general-purpose ball sports platform where we can simulate several ball sports with direct interaction. For example: basketball, volleyball, korfball, dodgeball, pass the ball, handball, netball, waterpolo, bowling, ... In this system we manipulate the ball based on the actions of the players, by doing so we are able to change reality.

1.2 Scenario of the system

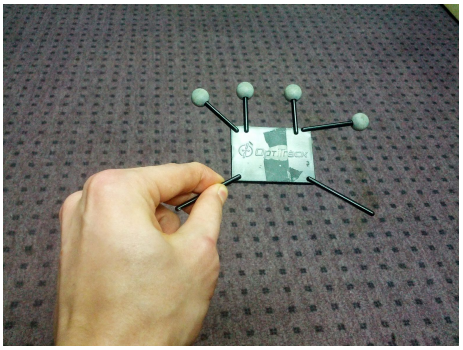
In this section we present a scenario of how players can use our system to play actuated basketball.

1.2.1 Setup

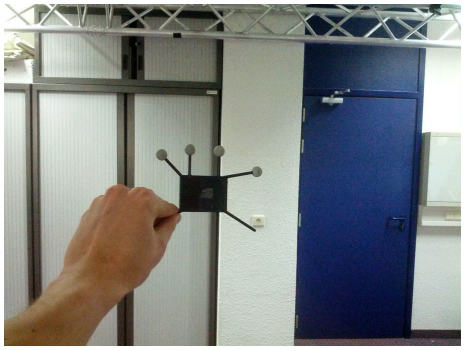
Before players can play the game they need to set up the system. This includes preparing the player equipment and the actuated ball, but also defining the playing field and the basketball hoops. Each player has a marker on his head and his wrist so the system can determine when a player holds the ball and where a player is located (see Fig 1.1a). The players are divided in two teams which compete against each other. The system is able to distinguish between the players once all the markers are identified. The players define the playing field size (see Fig 1.1b) and the location of the basketball hoops (see Fig 1.1c) by holding a marker on a position in the environment and pressing a button. All the the secondary sports attributes are only defined in the virtual world, for example invisible or projected. The actuated ball has markers attached to the frame in order for the system to control it based on the position and orientation (see Fig 1.1d).



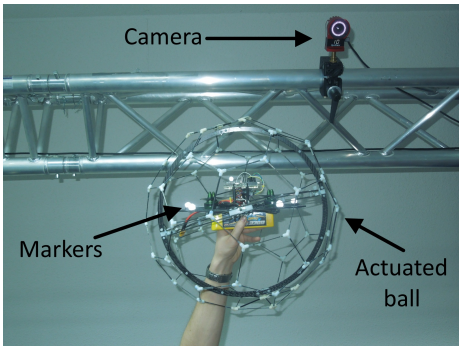
(a) Player with marker on head and hand



(b) Defining a corner of the playing field



(c) Defining the position of the basketball hoops

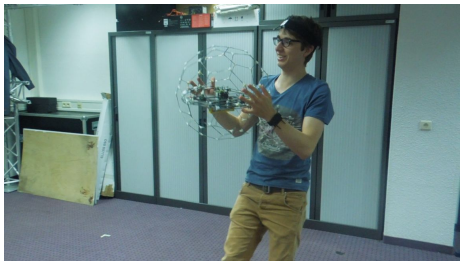


(d) Actuated ball tracked by tracking system

Figure 1.1: System setup

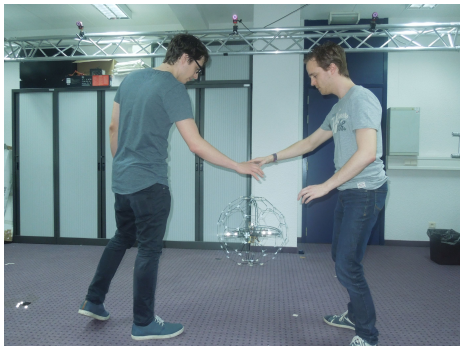
1.2.2 Playing the game

Once the system is set up the players can play the game. The normal rules of basketball apply and players can score points by throwing the ball inside the hoop of the other player (see Fig 1.2a). Players try to throw the actuated ball in one of the two basketball hoops (see Fig 1.2b). Dribbling with the ball is allowed in order to move with the ball (see Fig 1.2c). Players of the opposing team can try to block the ball (see Fig 1.2d). The system helps the players score a point if they made sufficient effort to throw the actuated ball in the basketball hoop, we will describe this in more detail in the next chapters. During the game the scoreboard displays the scores and the faults of each player.



(a) Player aiming towards a basketball hoop

(b) Player throwing the ball towards a basketball hoop



(c) One player dribbling with the ball while the other player tries to steal the ball

(d) One player trying to intercept the ball from the other

Figure 1.2: Gameplay

Chapter 2

Related work

2.1 Introduction

We introduce concepts of digital ball sports into the real world. We want to stimulate teamplay, incite players to move their body and create an challenging game experience. In this chapter, we first discuss existing exertion games in general while afterwards we categorise exertion games into special-purpose systems and general-purpose systems.

2.2 Exertion games

Exertion games are digital games that predominantly use physical effort as input. They engage players through physical as well as virtual spaces and objects. Researching exertion games is important because they motivate people to exercise and therefore have the potential to change sedentary lifestyles and associated health problems such as obesity.

In the paper Designing Sports [13] by Mueller et al. it is determined that we have a limited understanding how to create engaging exertion experiences. The reason for this is because exertion games they are too broad. They involve a wide variety of physical behaviours which happen everywhere each with their own specific tools.

The exertion framework described in the same paper related the framework [8] to exertion games. The framework helps with outlining and designing for compelling exertion experiences. The main goal is to describe how we can create more engaging experiences not optimize health benefits, however a more engaging experience will often result in health benefits automatically. The exertion framework acknowledges the fact that the human body plays a central role in exertion interactions. It proposes four lenses to provide a simple structure to reason about the human body and to promote creativity when designing for exertion games. The four lenses are the Responding Body, the Moving Body, the Sensing Body and the Relating Body (see Fig 2.1).

In the next sections we discuss the strengths and weaknesses of existing exertion games. The existing exertion games are divided into two categories based: special-purpose systems and general-purpose systems.

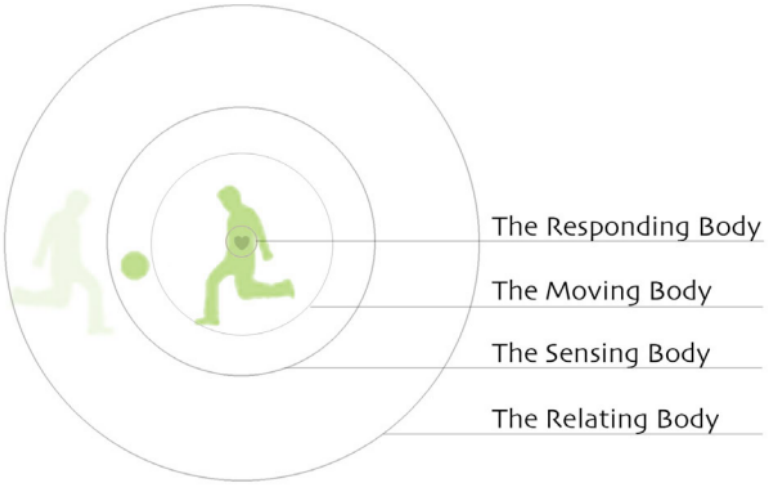


Figure 2.1: The four lenses in the exertion framework

Source: [13]

2.3 Special-purpose systems

Several existing exertion games are systems specifically made with one concept in mind, these systems are often close to the existing concept and therefore have a lot of physicality.

2.3.1 Jogging over a distance

Jogging over a distance [11] aims to support "social" joggers. A survey on jogging shows that joggers like to jog and talk with other joggers because it motivates them to run faster it makes the activity more pleasant. The survey identified there are two main challenges of social jogging. The first one is finding a jogging partner that runs at the same speed and the second one is finding a jogging partner that lives in the neighbourhood. However social jogging is impossible when joggers are geographically distant or when they have different physical capabilities which is exactly what this system tries to solve.

The system uses the relative difference in the heart rates of the joggers to determine if they run "next to each other". Each jogger hears the audio feed from the other jogger. However the audio feed is spatialized, the audio appears to come from different directions depending on the difference in relative heart rates. Therefore the spatialized audio tells the jogger where the other jogger is, for example if you are running "in front of" your jogging partner the sound will appear to come from behind you.

2.3.2 Jogging with a quadcopter

Jogging with a quadcopter [10] by Mueller et al. explores whether robotic systems can support the jogging experience. The system uses a quadcopter that accompanies a jogger by flying a certain distance ahead of the jogger on a fixed path at a constant speed.

Previously existing systems and technology that supported jogging differ from this system in various ways. One way they differ is the timing regarding the support. For example some mobile phone applications, such as Runkeeper and Endomondo, only provide data analysis after the jog. Smart watches focus on offering real-time feedback

during the activity, however this is only available when the jogger looks on the device. Therefore smart watches do not provide constant feedback or they cause fatigue when the user has to look on the device all the time.

Other systems support joggers constantly during the jog, such as "Zombies, Run!" [18] which uses headphones to deliver an immersive running game. Another example is "Jogging over a distance" that allows joggers that are geographically distant or with different physical abilities to jog together by using relative heart rate which is described in section 2.3.1.

Previously described systems support joggers by the means of audio or screen feedback, unlike JoggoBot [6], which provides support for joggers based on alternative technologies.

2.3.3 Remote impact

Remote impact [12] is a system that is inspired by combat sports. Real world as well as virtual combat sports require a lot of physical exertion and encourages the players to use their whole body.

This system allows two players to exercise combat sports on each others shadow while being geographically in two different places. Both players are in identical interaction spaces facing a touch sensitive wall. On this wall, both their own shadow and the other player's shadow are displayed. The goal of the game is to hit the opponent as many times as possible without getting hit yourself. When one player hits the other an audiovisual effect is displayed (see Fig 2.2). Unlike virtual combat sports or other exertion combat sports, such as some games on the Nintendo Wii, remote impact can feel force instead of measuring it based on movement.

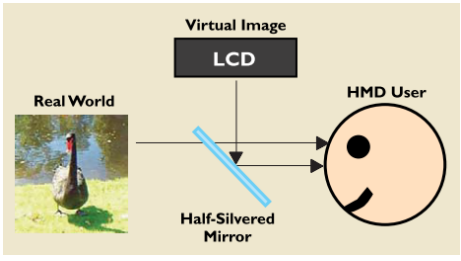
Players can execute very strong hits without the need for attributes, such as boxing gloves and without injuring themselves unlike in real world combat sports. Players can have and intense workout by punching rapidly or use the system to release stress. The system also trains their agility and reaction time, for example by dodging the other player.

Figure 2.2: Remote impact

Source: [12], https://www.youtube.com/watch?v=MJBFP9OATKg

2.3.4 ARQuake

Some systems remove the screen completely like ARQuake [17]. ARQuake extends the existing desktop game Quake into a system that allows to play augmented reality games outdoors. The system creates augmented reality by overlaying specific game elements of the desktop game on to the real world using a transparent head-mounted display (see Fig 2.3a). This allows the user to move in the physical world, and at the same time experience computer-generated graphical objects, such as power ups or monsters (see Fig 2.3b). It uses a physical real-life prop, a plastic gun with simulated recoil to shoot at the monsters.



(a) ARQuake head-mounted display

(b) ARQuake user view

Figure 2.3: ARQuake

Source: [17]

2.3.5 Table Tennis for Three

Table Tennis for Three [9] by Mueller et al. is a system in which three players can play table tennis against each other while all being at different locations. Each player has a table tennis table which has one side positioned vertically. On this vertical side the system displays a video stream of the other two players. On top of this the system projects a number of targets. The goal of the game is to destroy targets by hitting them three times. The targets are synchronised between the players, so other players can see when someone hits a target. Only the person that destroys a target by hitting it the final time gets the point.

In most exertion games there is indirect physical interaction with a tangible interface, therefore we can develop a certain set of skills. In this case we can learn to control the ball better and aim it more precisely. Also, people lack social interaction because they play alone or cannot meet each other. This system recreates the social and physical interaction with players at different locations. However this approach is very specific for table tennis and not generally applicable for every sport.

Figure 2.4: Table Tennis for Three

Source: [9]

2.3.6 Teqball

Teqball [1] is a combination of table tennis and football which uses a curved table, the "teqboard" (see Fig 2.5a). A tracking system is used to measure your skill level (see Fig 2.5b). Teqball is mainly a training tool for all football enthusiasts alike and aims to bring football into peoples lives. There is no direct physical contact between players of an opposing team and since it is not allowed to touch the equipment the risk of injury is very low.

[1] http://www.teqball.com/

(a) Two players playing a match of Teqball

(b) The tracking system that is used in Teqball

Figure 2.5: Teqball

Source: (a): https://www.youtube.com/watch?v=RD9d5VAUgYQ,
(b): https://www.youtube.com/watch?v=V1V9dYWX7IY

2.4 General-purpose systems

Create more general-purpose systems involves another way of representing the non realistic aspects in order to generalise them. Input sensor technologies make non realistic aspects virtual, for example seen through a screen, projected into the real world or invisible to the user.

2.4.1 Input sensor technologies

2.4.1.1 Nintendo Wii

The Nintendo Wii [2], released in 2006, was the first commercial system that made players use their whole body when playing a game. The Wii console uses handheld controllers, "Wiimotes", to support motor movement as gaming input. Players have to hold controllers in their both hands, the controllers determine the hand and arm movements the players make and the game uses this as input (see Fig 2.6a). The Wii comes with a sensor bar, which sends out infrared light. The controllers are equipped with accelerometers and infrared sensors. The accelerometers can determine gross movements while the infrared sensors can detect and translate signals send by the sensor bar. These infrared signals are used to triangulate the position of the Wiimote while pointed towards the screen.

To compensate for this, Nintendo offers users the option to buy attachments, such as the "Wii Motion Plus" which contains a dual axis tuning fork gyroscope and a single-axis gyroscope which can detect rotational motion.

The Wii can also receive input from the user using the Wii balancing board, which is mostly used in situations where the user has to balance his avatar. The Wii balancing board uses four pressure sensors to measure the user's weight and center of balance.

Players receive feedback mainly from the screen. Other forms of feedback include sound from the speakers or the controllers and even haptic feedback from the controllers.

Some Wii digital sports games use realistic attributes, such as a tennis racket or a baseball bat (see Fig 2.6b).

[2] http://wii.com/

(a) A boxing game on the Nintendo Wii

(b) Attributes used with some of the Nintendo Wii games

Figure 2.6: Nintendo Wii

Source: (a): http://www.videogamesblogger.com/2007/05/23/wii-sports-boxing-helps-rehabilitate-injured-boxer.htm,
(b): http://gamez.itmedia.co.jp/games/articles/0701/15/news093.html

2.4.1.2 Kinect

Microsoft's Kinect [3] was orginally released for the Xbox in 2010 and later released for the Windows platform.

Kinect uses external tracking therefore players are not required to hold a controller in their hands which lowers fatigue. External tracking offers the ability to track the movement of the player's complete body instead of only the upper body movements, which is an strength over the Nintendo Wii.

The absence of tangibility is both a strength and a weakness, players can join and leave easily because they do not need a physical controller. However on the other side the Kinect may be less useful for sports where the player has direct contact with an object, such as ball sports. External tracking also means no haptic feedback from controllers. Players receive feedback mainly from the screen and sound from the speakers.

[3] https://www.microsoft.com/en-us/kinectforwindows/meetkinect/default.aspx

Figure 2.7: A balancing game on the Kinect

Source: http://www.kotaku.com.au/2010/07/the-whole-world-looks-silly-playing-kinect/

2.4.1.3 Imaginary Reality Gaming: Ball Games Without a Ball

Imaginary Reality Gaming [3] is a system that features real world sports without an actual visible ball. To accomplish this, the system uses a virtual ball, creating screenless augmented reality. The system uses a webcam based tracking system to track the players but also the playfield boundaries, baskets, and power-up locations. Players have a marker on their head which is captured by a webcam to indicate their position and orientation. To track the hand movements of the players accurately hand-worn accelerometers are used.

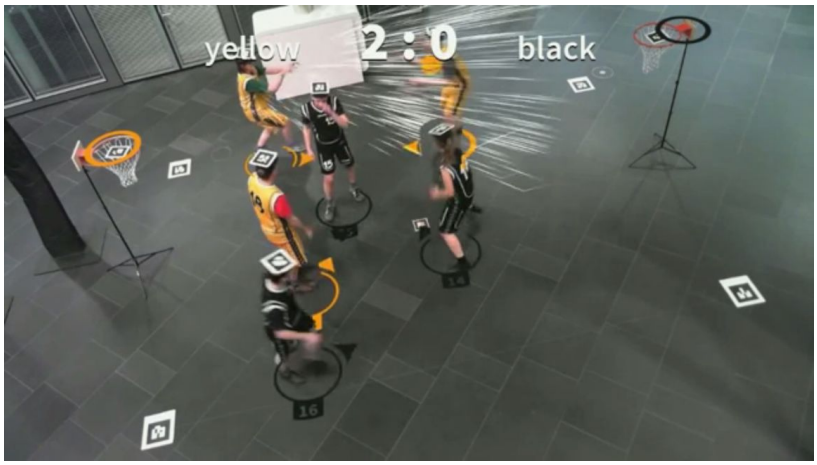


Figure 2.8: Imaginary Reality Gaming

Source: [3]

This info is given to a modified physics engine, *the quantum engine*, which computes 500 different versions of the same ball. The physics engine uses particles to represent

each version of the ball which is displayed in the debug view (see Fig 2.8). The black and orange indications and the particles are virtual. The quantum engine evaluates each version of the ball and aggregates the probabilities of each outcome. More traditional approaches would prevent players from passing the ball to a player that is occluded by an opposing player while using this technique the player still has a chance of receiving the ball (see Fig 2.9). A big difference compared to exertion games is that there is no physical interaction with the ball.

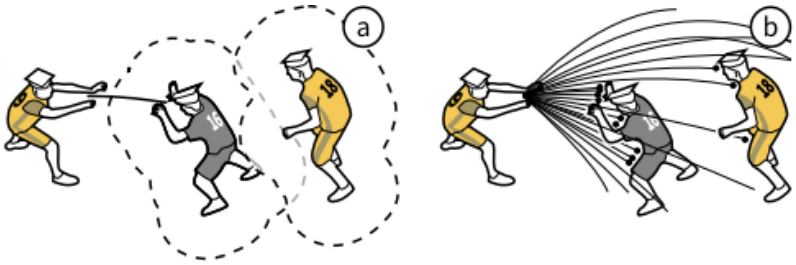


Figure 2.9: Particle system

Source: [3]

Since players cannot physically touch the ball or directly interact with it, they interact using gestures. These gestures resemble real life actions similar to actions used in the specific sport they are playing. For example throwing a ball into a basket hoop in basketball. Players try to determine the position of the virtual ball by watching the gestures of the other players or auditory feedback. Auditory feedback mainly includes moments when the ball makes contact with a player or an object. The system combines the ability to have virtual game elements, such as power-ups and non-physical objects with direct player interaction. One of the strengths of the virtual reality aspect is that the games are not limited by what is possible according to the real world physics. However creating a general-purpose gaming platform that can be used for many ball sports using this approach is cumbersome. If we want to play or implement a new sport we need to manually set up new secondary sports attributes which is a hassle. A second weakness is that players cannot use the ball skills they acquired over the years because there is no physicality.

2.4.1.4 Mano-a-Mano

Mano-a-Mano [4] is a system that focuses on dyadic projected spatial augmented reality which allows two users to interact with a shared virtual scene and each other while faced towards each other.

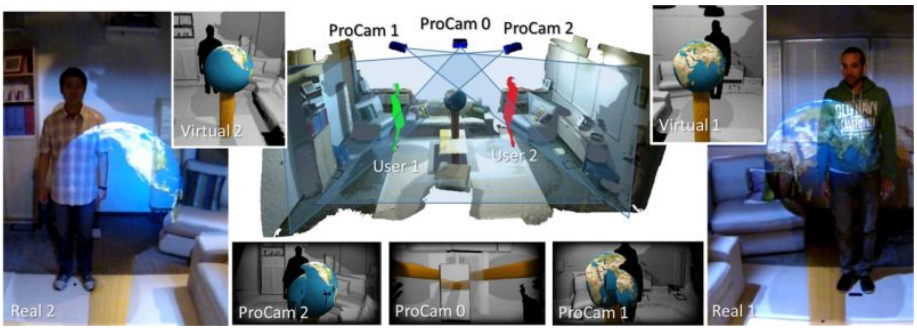


Figure 2.10: An example application of Mano-a-Mano which projects a virtual ball in the real world.

Source: [4]

The system consists of three projectors each paired with a kinect (see Fig 2.10). The kinect provides precise surface geometry which is needed for dynamic projection mapping. Two of the projector and kinect pairs are mounted above the two users facing the other projector. The third projector and kinect pair is facing completely downwards in order to cover the area between the areas covered by the other two.

One of the many example applications projects a virtual ball in the real world which the players interact with. Because the ball is projected in the real world, players are able to see the facial expressions of other players and therefore engage more with each other instead of observing the screen.

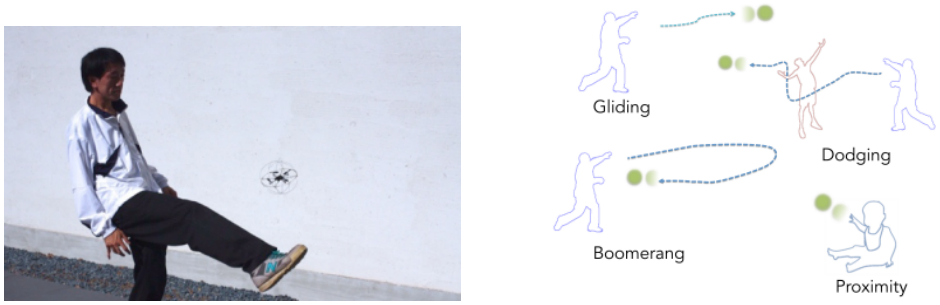
2.4.2 Actuation technologies

Actuation technologies make non realistic aspects physical, for example by making a ball react on the gestures of the user.

2.4.2.1 HoverBall

HoverBall [14] is a system that aims to create augmented sports using flying robotic balls. Balls are the most popular equipment for sports. However they are subject to gravity and the laws of physics, which could prevent certain groups of people to play with them. For example, balls often move to fast for children and elderly people or people with physical disabilities have problems interacting with balls.

HoverBall uses a flying ball to remove the subjection to gravity (see Fig 2.11). The flying robotic ball is a quadcopter that can be remotely controlled with a ball-shaped case around it. By removing these limitations from the ball, we can come up with new interaction methods. Traditional ball sports are limited to traditional interaction methods, such as throwing the ball. A new set of interaction methods includes proximity or remote pushing or pulling the ball. Another strength is that this system facilitates ball control for weaker players. The ball behaviour will change depending on the players. This is achieved by correcting the trajectory of the ball or adjusting the velocity of the ball. Players can compete seriously, which contributes to a more exciting and amusing game experience.



(a) HoverBall **(b)** HoverBall gestures

Figure 2.11: HoverBall

Source: [14]

2.4.2.2 Sphero

Sphero [4] is a robotic ball that can move autonomic and can be controlled from a distance using a smartphone application. The ball movement is purely mechanic, therefore it can only roll move on the ground. It can adjust its position and orientation, which can change the trajectory of the ball as long as it is on the ground. Players can interact with the ball by pushing, rolling, moving it around.

Figure 2.12: Sphero

Source: http://www.cnet.com/pictures/tech-toys-for-big-kids/5/

2.4.2.3 Motion-Controllable Ball and TAMA

Motion-Controllable Ball [7] and TAMA [16] aim to make ball games more enjoyable and exciting by changing the trajectory of the ball during flight (see Fig 2.13).

[4] http://www.gosphero.com/sphero/

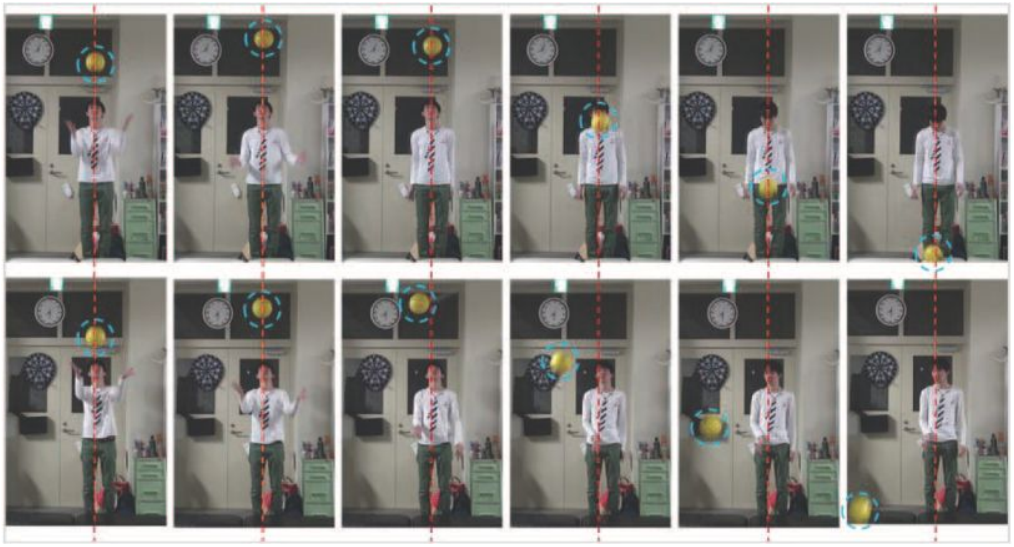
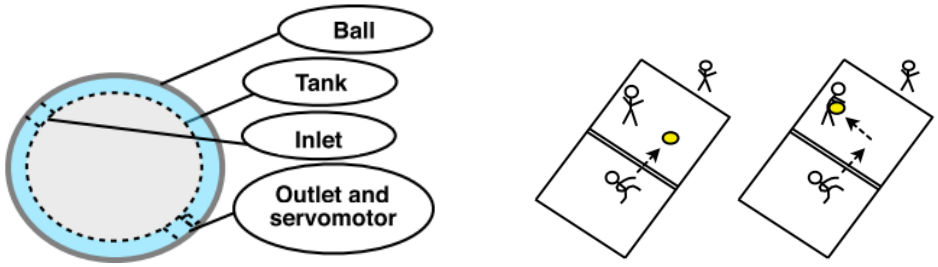


Figure 2.13: TAMA

Source: [16]

In both systems the trajectory of the ball is adjusted by the ball itself using a gas-jet system (see Fig 2.14a). Dynamically motion-controllable balls allow for new interactions such as adjusting ball trajectory to goal or targets or controlling the speed of the ball or the distance it covers.



(a) Motion-Controllable Ball

(b) Concept of dodgeball with a change of the ball vector

Figure 2.14: Motion-Controllable Ball

Source: [7]

These systems can be used in sports to support players with a different physique or skill level, for example adults and children, to compete with each other. Another use case is making the match more exciting and tactical. In baseball, for example, it allows for new breaking balls and controlling the ball speed or distance of flying. It also introduces more variety into the game, the same set of actions will not result in the same game.

Another example is dodgeball, the fact that the vector of the ball can change can lead to unexpected results. In dodgeball the players participate both inside and outside the playing field. Players outside the playing field can be reached by throwing the ball through the inside part of the opposing team. If the direction of the ball suddenly changes, it can hit an opposing player (see Fig 2.14b).

2.4.2.4 PhotoelasticBall

PhotoelasticBall [15] describes the development of a force vector sensor which allows balls to recognize the position and direction of the force applied to them. In more static setups it is possible to use joysticks and buttons, which are more traditional input methods, to control an application. For balls on the other hand these traditional input methods are not suited for most ball games. Balls can rotate while flying, combined with the fast-paced nature of these ball games there is not enough time to search for the button or stick. This system allows the user to use a ball as an input device even when it frequently changes position and orientation.

Figure 2.15: Concept of PhotoelasticBall

Source: [15]

2.5 Conclusion

Many existing exertion games, such as these discussed in section 2.3, for example Remote impact and Table Tennis for Three, are systems made specifically for a certain sport or game concept. Therefore it is often not possible to use the system to implement a different concept without making large modifications to the existing system. Another weakness is the need for different attributes, for example in Table Tennis for Three (see section 2.3.5) we always need a table tennis table, balls and paddles. Special-purpose systems are very close to the real world sports with the addition of some virtual aspects, therefore they have a lot of tangibility like the real world sport itself.

A common solution to create more general-purpose systems is to make the non-realistic aspects virtual, such as the systems in section 2.4.1. The strength of these systems is the possibility to play multiple games with the same system, often without the need for reconfiguring the environment. These systems often include unrealistic physics and power-ups. However the majority of these general-purpose systems also have some weaknesses compared to their real world counterparts. One weakness is that there are often limited feedback channels, in many cases visual feedback is the dominant form of feedback, for example the Nintendo Wii and Microsoft Kinect (see sections 2.4.1.1 and 2.4.1.2). Therefore players are required to focus on the visual feedback for the majority of the feedback which results in decreased social interaction between players compared to the real world ball game. A second weakness is the lack of physicality since the majority of the aspects are virtual, for example the Imaginary Reality Gaming system (see section 2.4.1.3). Players that interact often with physical attributes, such as balls, will develop various skills that are specific to the game or interaction. Without physicality players are unable to use the precise coordination and control that they developed with physical attributes over a lifetime

To regain this physicality without losing the non-realistic aspects, such as unrealistic physics, we use actuated balls, such as HoverBall and Sphero (see sections 2.4.2.1 and 2.4.2.2). The fact that we can control the ball while it is present in the real world maintains both the unrealistic physics and the tangibility.

Chapter 3

Conceptual design of actuated ball games

3.1 Introduction

Our system is a general-purpose ball sports platform which allows multiple games to be deployed in a similar room environment however, as discussed in chapter 2, existing exertion games are often quite specific. A solution to this problem is to make a more general-purpose system by making all non-realistic aspects virtual which offers the advantage of unrealistic physics. However this solution also has weaknesses, such as the lack of physicality and less sociability. Actuated balls are balls that move autonomic and are controlled from a distance. Digital sports games and their real world counterparts each have their own strengths and weaknesses. Our general-purpose ball sports platform which we describe in more detail in this chapter combines several strengths of both categories into a single system by using actuated balls.

3.2 Analysis of digital and real world ball sports

In this thesis we combine the strengths of digital ball sports and their real world counterparts in a general-purpose ball sports platform by using actuated balls.

Digital ball sports have the weakness that there is often a single visual feedback source, such as a screen, which provides the majority of the feedback. Therefore players are required to focus on the screen which prevents them from observing the facial expressions of other players and often results in decreased social interaction. Since in most cases this single source is the screen we want to remove the screen entirely and provide audio feedback. A second weakness is the fact that none of the attributes are physical, therefore there is no physical interaction possible. Physical interaction is important because players can use the various skills, such as precise coordination and control, they developed over a lifetime. Because physical interaction is not possible in digital ball sports players are unable to use their already acquired skills or develop their skills further.

However digital ball sports also have some strengths we want to preserve, such as unrealistic experiences. There is no subjection to gravity, therefore unrealistic physics are possible, for example gravity like on the moon, inversed gravity or a ball bouncing of a virtual wall. Power-ups are often used to reward players or give players an advantage over others. The ball characteristics such as size, weight and texture are changed easily. The second strength is the fact that there is no need for secondary sports attributes, since

all secondary sports attributes are virtual. Therefore we easily switch between games or add new games without the need for different physical attributes, it is a virtual general-purpose system. On the flipside however, the lack of physicality often impedes the use of a wide variety of skills that we normally use when interacting with real physical balls, such as juggling, dribbling,...

In contrast to digital sport games, real world sports encourage social interactions between players because players are not focused on a screen. All the attributes are physical thus there is physical interaction which allows players to develop certain skills. The strengths of digital sports games, such as unrealistic physics and virtual attributes, however are absent in the real world. Objects are subject to gravity and realistic physics and the need for physical attributes complicates the creation of a true general-purpose system.

3.3 Concept of the general-purpose ball sports platform

By combining several of the concepts discussed in section 2 we introduce the strengths of digital world sports into the real world, thus eliminating the weaknesses we discussed above. The core of our system consists of a tracking system and an actuated robotic ball. The actuated robotic ball allows us to preserve the physicality and tangibility of the ball and allow for unrealistic experiences at the same time, which enables the system to change reality. To create a general-purpose system in the real world we render all non-realistic aspects virtual which allows for switching between multiple games, power-ups, virtual walls, etc. Because the system operates in the real world players are not focused on a single source of feedback, such as a screen.

3.3.1 Interaction

Players interact with the actuated ball in the form of direct interaction by using their hands. Players hold the ball in their hands and release it afterwards. Releasing the ball while there is no movement results in the ball dropping on the ground. If the ball is released while moving it is launched into the air.

When players interact with balls during sports they develop skills that improve ball coordination and control of the ball in general. For example, dribbling, applying spin to the ball, smashing the ball over a volleyball net or passing a ball to another player.

Dribbling in basketball is a way of moving the ball by bouncing it continuously on the floor with one hand while running on the playing field. It requires a certain ball coordination in order to not lose the ball while running.

Applying spin to the ball is important in sports such as bowling and basketball. In bowling a spinning ball is used in order to knock down all the pins more easily because it hits the pins with more force. Another example is basketball where the player applies spin to the ball and aims for the top corner in order to make the ball approach the hoop more softly and increase the chance of scoring.

Smashing the ball over a volleyball net requires the player to jump at the right time. The player preferrably hits the ball in such a way that it lands immediately on the ground.

Passing a ball to a team player, be it in basketball, volleyball, water polo, is hard due to the fact that the ball should not be intercepted by an opposing player or incorrectly received by the team player.

3.3.2 Engaging in the gameplay

The system decides when and how to engage in the gameplay based on the game logic, which differs for every game. First we give examples of when and how the game decides to engage in the gameplay, afterwards we will clarify these examples by using them in real life scenarios.

Examples of when the system decides to engage in the gameplay are rewarding and punishing players, helping players reach a goal or balancing player skills. One possible condition when to reward players is whether all players are included in the game, for example by for passing the ball around. Punishing players happens for example when breaking the rules or incorrectly handle the actuated ball. Helping players reach a certain goal, for example throwing a ball in a basketball hoop. Balancing player skills allows players with different capabilities to compete against each other by applying handicaps or helping the less skilled player.

Examples of how the system engages in the gameplay are adjusting the trajectory, adjusting the movement speed of the ball or changing the required force to throw the ball. Adjusting the trajectory of the ball is used to help players reach a goal or when moving around a player to avoid him. Slowing down the actuated ball or making the ball harder to throw is used to apply a handicap to a more skilled player and creates an effect similar to when the ball is thrown under water. By making the ball easier to throw we realise the effect like throwing the ball on the Moon.

In basketball the system encourages the players to include other players by counting the number of passes between players. When enough passes are given the system rewards the players by helping to score a point by adjusting the trajectory of the ball towards the basketball hoop.

When children want to play bowling against their parents there is a clear difference in physical capabilities between the two groups. In order to allow these two groups to compete, the system makes it easier for the children to hit the pins in such a way to knock down as much pins as possible. Another option is to protect the ball from going into the gutter, assuring that the ball always hits the pins.

In dodgeball the goal is to hit players from the opposing team. Player participate inside and outside the playing field and it is possible to reach players outside the playing field by throwing the ball through the inside part of the opposing team. The system allows the player who threw the ball to adjust the trajectory of the ball in the air by using gestures. If the direction of the ball suddenly changes, it can hit an opposing player unexpectedly.

In the current implementation we focus on helping the player achieve a goal, in this case throwing a basketball into a basketball hoop.

3.3.3 Unrealistic experiences

The system introduces unrealistic experiences in the real world by using an actuated ball, effectively changing reality.

Players are able to play basketball with decreased gravity, as if the game takes place on the Moon.

The ball bounces off virtual walls, which can for example be used to recreate a game of squash where the players hit the ball with their hands instead of with rackets.

In the case of an injury or when a game rule is broken, the ball hovers high in the air. This prevents players from continuing the game and is a clear indication that the game is interrupted. It is possible to alternate the speed of the actuated ball, which makes the game more interesting.

Other unrealistic experiences include fluids, for example players can play water polo. The system increases the resistance applied to the ball once the ball is under water, thus making it harder to move the ball.

3.3.4 Feedback

To allow for a system that can deploy different games in a similar environment we create a general-purpose system in the real world. To be able to switch between multiple games we render all non-realistic aspects virtual. Non-realistic virtual aspects include virtual walls, power-ups and secondary sports attributes. These can be invisible to the user or projected in the real world.

The system compensates for the diminished visual feedback by providing audio feedback. Audio feedback includes the collisions of the actuated ball with the virtual objects, such as the floor, walls and secondary sports attributes. A different sound is played for each collision. Another form of audio feedback is playing a sound based on the result of the actions of the players which can either be positive or negative feedback. For example in basketball we want to reward players when they play together as a team based on the number of passes they make. If the number of passes is higher than a certain number the system plays a sound indicating that the players are rewarded for their actions. Examples of negative feedback include breaking the rules or hitting the actuated ball too hard which damages the system setup.

Feedback after the activity includes analytics of the played game, the number of goals, errors, passes, etc.

3.4 Conclusion

In the first part of this chapter we discussed the strengths and weaknesses of digital and real world ball sports. We concluded that the strengths of digital ball sports compensate for the weaknesses of real world ball sports and vice versa. In the second part we describe our system which combines the strengths of digital ball sports games with those of real world ball games. We achieve this by introducing the strengths of digital ball sports games into the real world.

Chapter 4

Technical requirements of actuated ball games

4.1 Introduction

Our system introduces unrealistic experiences in the real world by using an actuated ball. We control the actuated ball which allows us to change how it behaves and adapt the trajectory of the ball. In order to control the actuated ball we need to acquire the position of the actuated ball for which we use the tracking system.

4.2 Tracking system

We want to be able to react to the actions of the user, for example by changing the trajectory of the ball or changing the speed of the ball. In order to do this, we need to control the actuated ball accurately. To achieve control of the ball we need to be able to move the actuated ball in all directions for which we need to know the position and orientation of the ball at all times. The positioning needs to be as fast as possible and accurate and precise enough in order to control the actuated ball. Timing is important, for example if we drop an object from one meter high without applying any force it will land on the ground in less than half a second. Therefore it is required to receive position and rotation updates at least every 10 milliseconds, however faster if possible.

The tracking system is used to provide sub millimetre absolute positioning and orientation of objects, such as the actuated ball and the players. The main reason why we use a tracking system is to control the actuated ball and correct it's movements based on the current position and orientation of the actuated ball. Several markers are placed on the actuated ball in order for the tracking system to track it (see Fig 4.1).

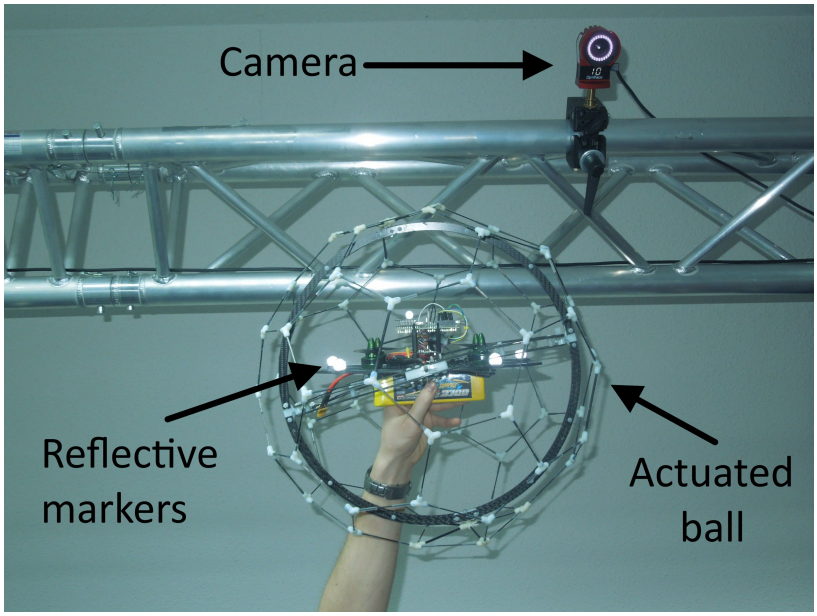


Figure 4.1: Actuated ball with markers

4.3 Actuated ball

The actuated ball needs to be able to move in all directions as fast as possible. It needs a minimum flight time of the duration of a game therefore the capacity of the battery needs to be high enough. In order to not consume a lot of energy the actuated ball needs to be light. The actuated ball has to be as quiet as possible to not disturb the game or the audio feedback. The ball needs to be strong enough to withstand forces, it needs to be safe and user friendly.

In order for the system to engage in he gameplay, the actuated ball must be able to react properly to an adjustment of the trajectory, for example when trying to reach a goal or evading objects. The ball must stay stable when thrown to create a smooth game experience. The movement speed of the actuated ball must be adjustable on the fly. The system must be able to control the resistance of the ball when thrown. Decreasing the resistance of the ball when thrown simulates a lighter ball or aims to help the player while increasing the resistance accomplishes the opposite.

Some of these requirements are also needed in order to introduce unrealistic experiences in the real world. An adjustment of trajectory is needed in order to change the gravity of the ball or to simulate the reaction of the ball under water the system needs to increase the ball resistance.

4.4 Code framework

The system needs to be able to accurately control the actuated ball. More specifically the actuated ball needs to move as fast a possible to the destination. The system receives the position and orientation of the actuated ball from the tracking system and corrects the position based on the destination in a control loop.

It must be possible to implement multiple game concepts in the system. Creating a new game concept or adapting an existing one has to be easy. Therefore the dimensions of the playing field and the positions of secondary sport attributes need to be easily adjustable. The game logic must be able to enforce the rules of the game and determine

the reaction of the ball based on the actions of the players. In order to react quickly to the actions of the players the system needs to be able to predict the trajectory of the ball, for example by adjusting the trajectory or changing the speed of the ball.

4.5 Conclusion

In this chapter we described the system architecture, the components that are required to create the system. We illustrated how they interact with each other in order to introduce unrealistic experiences into the real world by actuation. The tracking system is used to determine the position of the actuated ball and the players. The actuated ball needs to move as fast as possible to the destination. The code framework combines these two components and allows to create new game concepts easily. The actuated ball is controlled by the code framework which constantly updates and corrects the position of the actuated ball based on the information received from the tracking system in a feedback loop.

Chapter 5

Implementation

5.1 Introduction

We create a general-purpose ball sports platform by implementing the concepts of actuated ball games, discussed in chapter 3. The advantages of real world sports and digital sports games complement each other in the real world. Our system introduces non-realistic aspects into the real world, such as unrealistic physics and power-ups. It maintains the flexibility of traditional virtual games while using a physical ball to preserve the skills humans developed with these balls over a lifetime.

In this chapter, we elaborate on how we created our system, we discuss all the different parts, such as the tracking system, the actuated ball and the code framework that allows players to play actuated ball games using the general-purpose ball sports platform. Fig 5.1 provides an overview of the system and displays how the different parts communicate.

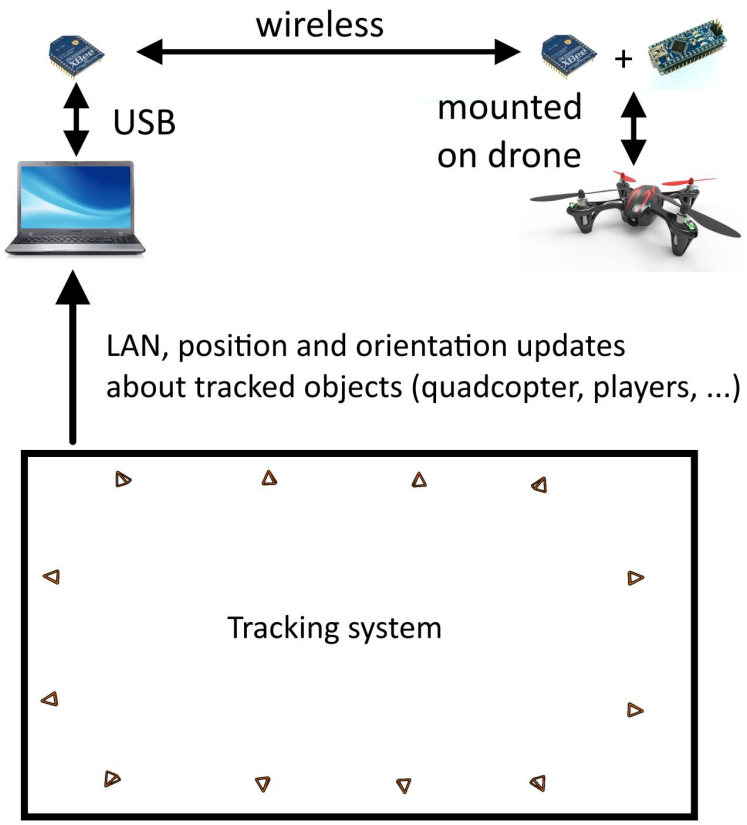


Figure 5.1: Overview of the system

5.2 Tracking system

The first component of our system we discuss is the tracking system which tracks the position of the ball and the players. Our system uses an Optitrack tracking system, which is an optical tracking system with twelve cameras to acquire sub millimetre absolute positioning and orientation (see Fig 5.2). The tracking system runs at 120 frames per second, therefore every 8.3 milliseconds we receive an update about each tracked object.

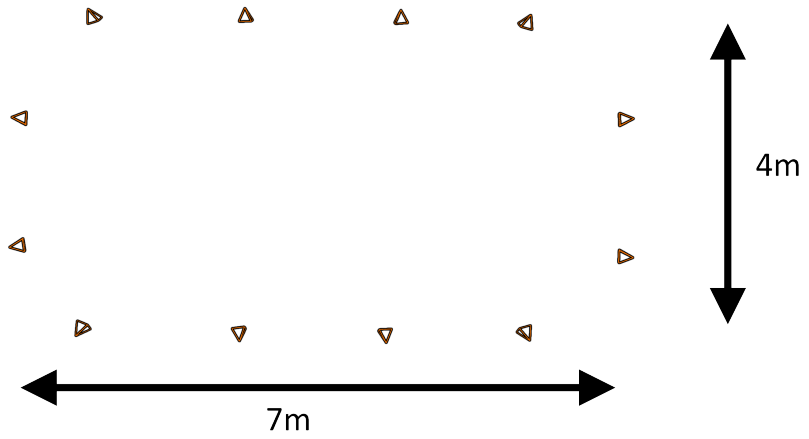
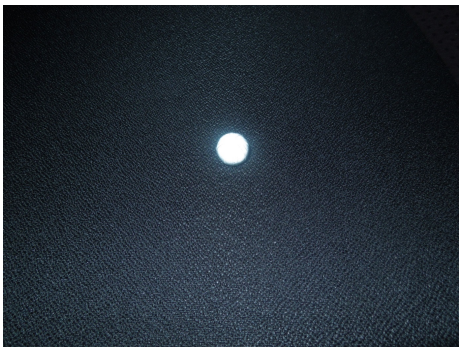


Figure 5.2: Top down view of the optitrack system

The main reason we use the tracking system is to control the actuated ball and correct it's movements, for this we need it's position and orientation. The infrared lights mounted on the cameras emit infrared light which reflects on the passive retro-reflective markers (see Fig 5.3a and 5.3b). Infrared light is invisible to the human eye because it operates on a wavelength of 700 nm to 1 mm. Based on the reflected infrared light captured by the cameras the tracking system triangulates the position of the markers. Triangulation requires a single marker to be in the view of at least three cameras for the tracking system to be able to determine it's position. It is possible for markers to be occluded from the view of a camera if a solid object is located between the marker and the camera.

(a) Passive retro-reflective marker



(b) Passive retro-reflective marker, picture taken with flash, visible reflection on the marker

Figure 5.3: Passive retro-reflective markers

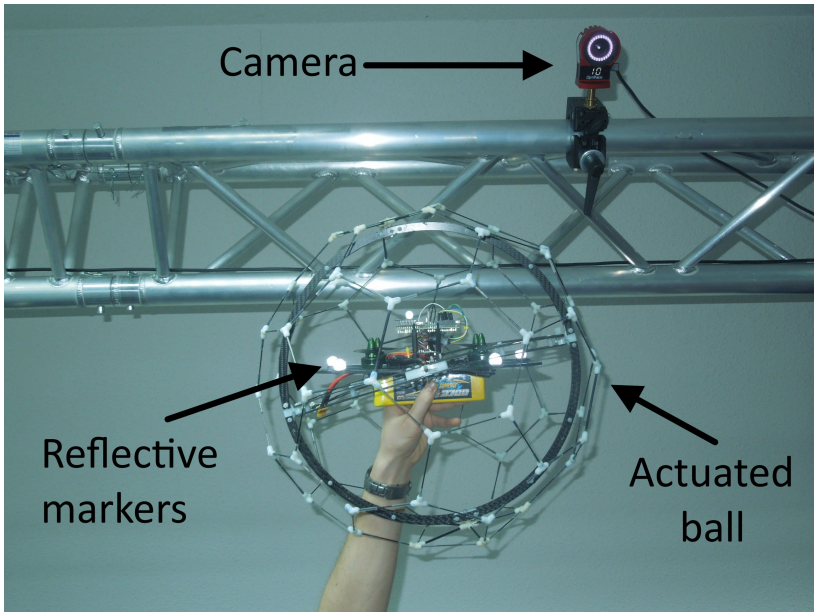
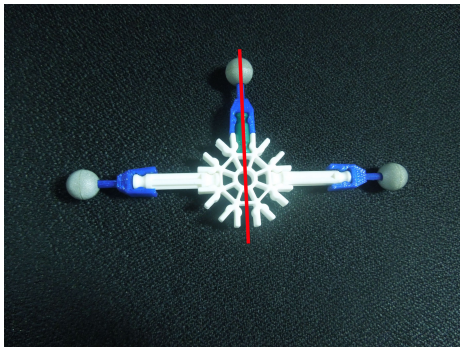


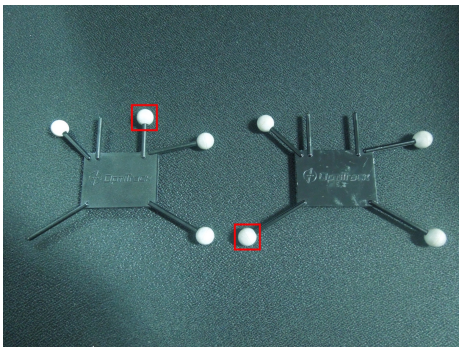
Figure 5.4: Actuated ball with passive retro-reflective markers

The retro-reflective markers are attached on objects we want to track, for example with double-sided tape (see Fig 5.4). The placement of the markers on the object, thus the relative distance between the markers, forms a pattern in which it is important to make sure there is no ambiguity between markers placed on the same object or between tracked objects. Ambiguity between markers placed on an object is occurs when all markers are coplanar and the placement of the markers is symmetrical along

an axis (see Fig 5.5a). Ambiguity between tracked objects occurs when objects have the same subpattern of markers. For example if certain markers on tracked objects are occluded and the remaining visible pattern is the same, the tracking system is unable to distinguish between these objects (see Fig 5.5b). Therefore it is important to create as many unique patterns as possible without any matching subpatterns.



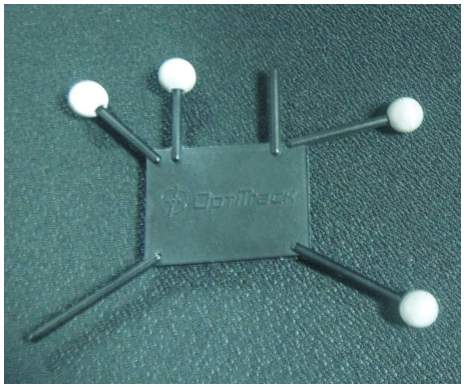
(a) Unable to determine the rotation of the object around the specified axis



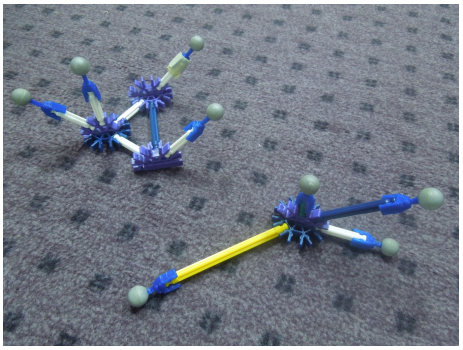
(b) If the markers indicated with a red square are not tracked the system is unable to differentiate between these tracked objects.

Figure 5.5: Marker ambiguity

The default Optitrack markers (see Fig 5.6a) have 6 places where markers can be attached which does not offer a lot of possible patterns. In our system we need two markers for each player, one attached to the head of the player and one to the wrist. While it it possible to create eight unique markers there is a lot of overlap between the markers and if markers are occluded, the system confuses several players. Therefore, we 3D-print extensions that fit on Knex[1] and can hold a marker, in order to increase the diversity (see Fig 5.6b).



(a) Default Optitrack marker, limited number of patterns



(b) Custom marker made with knex, more solutions possible

Figure 5.6: Knex markers

5.3 Actuated ball

In order to realise unrealistic physics in the real world we use an actuated ball which is remotely controlled by the system. The actuated ball of our choice is a quadcopter, which

[1] http://www.knex.com/

we control, with a ball-shaped case around it. Our quadcopter is an unmanned aerial vehicle (UAV or drone) which is controlled from a distance. Generally a quadcopter is controlled manually by a person with a controller, however in our case it is controlled by the system.

5.3.1 Quadcopter controls

Quadcopters have a standardized control mechanism that consist out of 4 variables used to steer the copter: *roll*, *pitch*, *yaw* and *throttle* (see Fig 5.7).

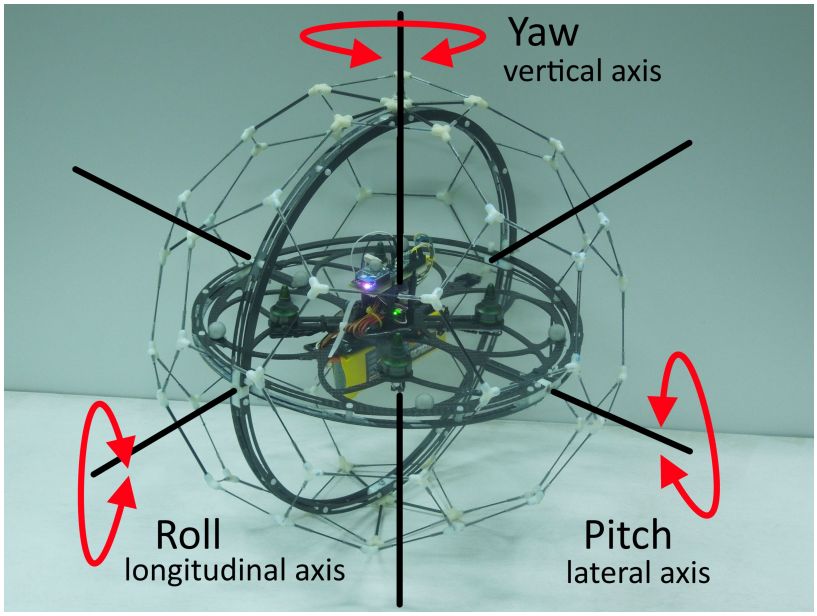
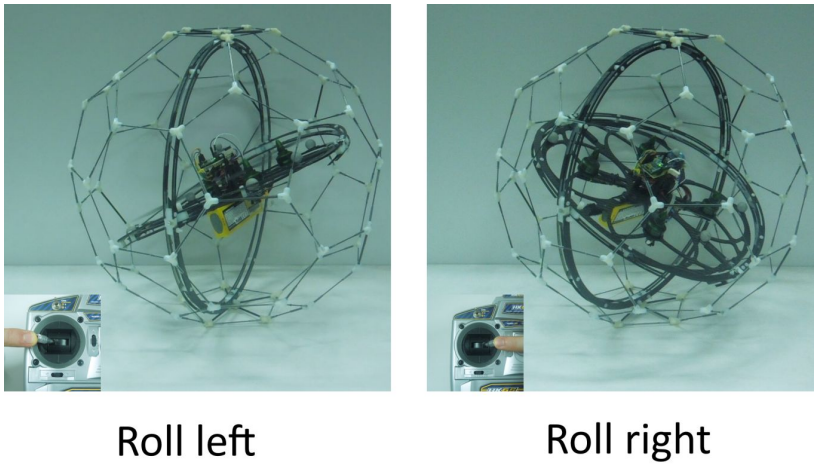
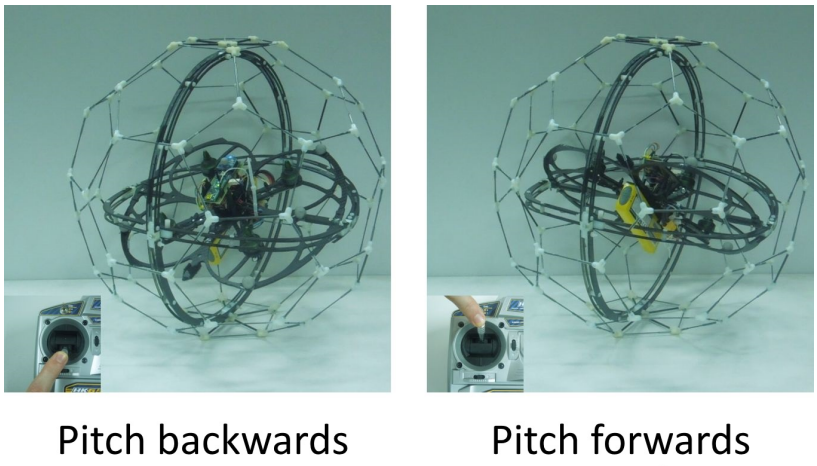


Figure 5.7: Yaw, pitch and roll visualised

Roll rotates the quadcopter around the longitudinal axis (see Fig 5.8a), while *pitch* rotates the around the lateral axis (see Fig 5.8b). *Yaw* rotates the quadcopter around the vertical axis which is not crucial for our implementation. Finally *throttle* is the amount of power we let the propellers produce, which determines the altitude and velocity of the movement.



(a) Roll makes the quadcopter go left and right



(b) Pitch makes the quadcopter go forwards and backwards

Figure 5.8: Quadcopter controls

5.3.2 Toy actuated ball

Our first actuated ball is the SkyWalker (see Fig 5.9) a toy quadcopter with a ball shaped case around it that rotates around 1 axis. The quadcopter is very light, it only weighs 84 gram, and it is hard to control manually using a remote control. The system is able to control it more accurate than a human could, however once we started using the quadcopter we concluded that it is not completely suitable for our needs. The quadcopter is 24 by 21 by 21 cm.

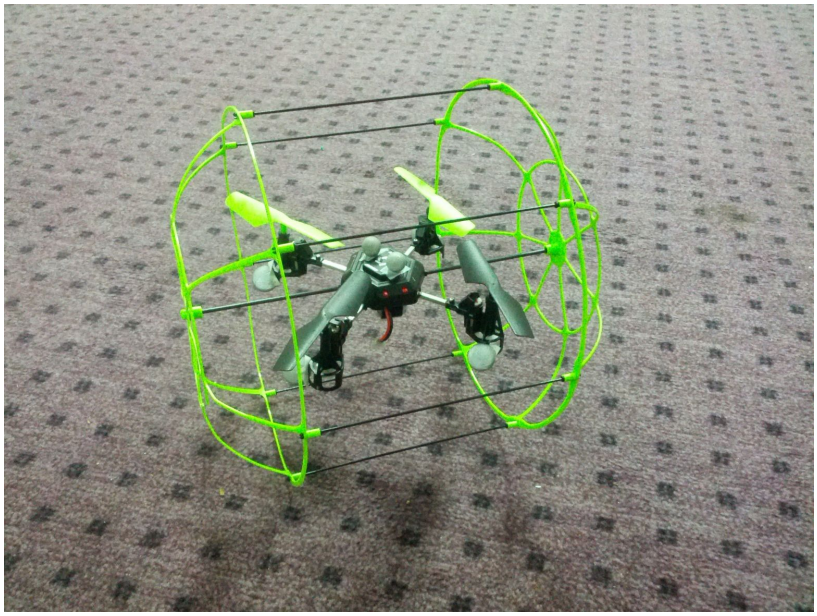


Figure 5.9: Toy quadcopter

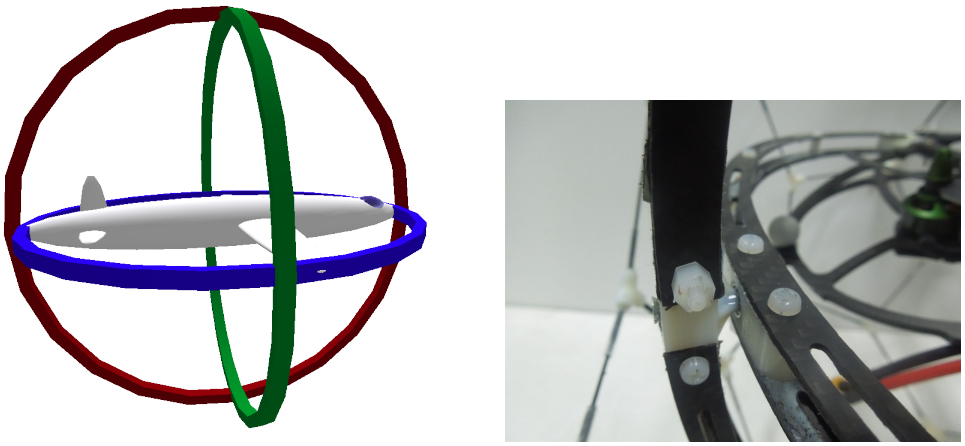
If a player pushes the quadcopter perpendicular to the axis the case rotates around the quadcopter while the quadcopter itself remains horizontal. However if the quadcopter is pushed from another angle the quadcopter itself rotates as well as the case as a result the quacopter rotates and loses height.

Other concerns include short flight time, high throttle required to hover and the high impact of the battery capacity on the flight experience. A flight time of maximum six minutes is a problem because players must not be limited in their playtime.

The required power to hover about 1 meter above the ground which is around 80% of the maximum value, however the higher the quadcopter has to fly the more the required power increases, therefore the quadcopter is unable to hover at any height above a certain limit. The battery capacity impacts the flight experience a lot, for example if the battery is full the quadcopter lifts off fast, however after a minute of hovering in the air the quadcopter lifts off much slower. Other quadcopters also have these two last problems, however not to this extent. To solve these problems we build a custom quadcopter which we discuss in the next section.

5.3.3 Custom actuated ball

The design of the custom actuated ball is made by Oeve Geypen and the 3D printing and construction is done by Raf Ramakers with my assistance. The actuated ball is a quadcopter mounted into a sphere-shaped case. It weighs around 900 gram, is 45 cm in diameter and is able to stay in the air for about ten to twelve minutes. The actuated ball is mainly constructed out of carbon, which is a very high tensile strength and low weight. The remaining parts of the construction are 3D-printed. Since the quadcopter functions as a ball the case is sphere-shaped and has to be able to rotate in any direction while the quadcopter stays horizontal. To stabilise the quadcopter we use a system with three gimbals (see Fig 5.10a).



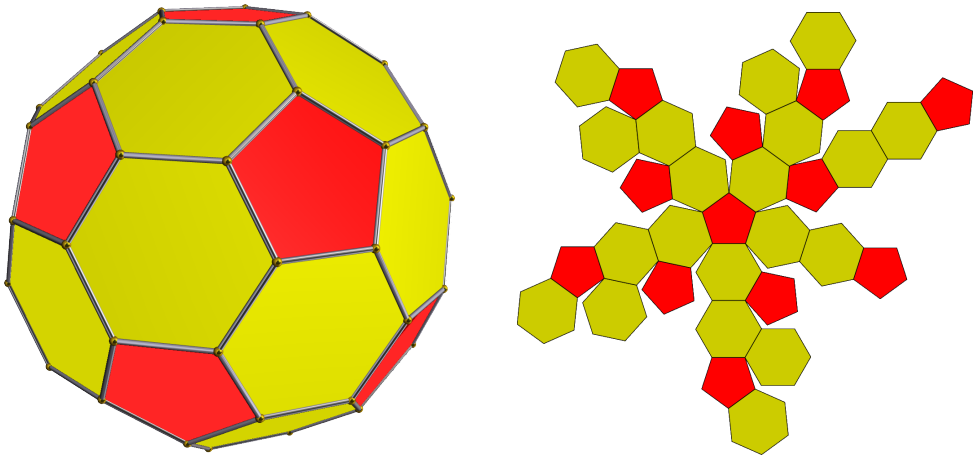
(a) Gimbal principle

(b) Two gimbals connected with a ball bearing

Figure 5.10: Gimbal principle used in the quadcopter

Source: (a): http://en.wikipedia.org/wiki/Gimbal_lock#/media/File:No_gimbal_lock.png (made by MathsPoetry)

The case is attached to the outer gimbal and the inner gimbal is the quadcopter itself by doing so the case rotates freely while the two gimbals make sure the quadcopter stays horizontal. The gimbals are connected to each other with ball bearings (see Fig 5.10b), making them a passive system based on the laws of physics.



(a) Quadcopter case in 3D

(b) Quadcopter case in 2D

Figure 5.11: Quadcopter case

Source: http://en.wikipedia.org/wiki/Truncated_icosahedron

Source: (a): http://en.wikipedia.org/wiki/File:Truncated_icosahedron.png (made by Tomruen), (b): http://en.wikipedia.org/wiki/File:Truncated_icosahedron_flat-2.svg (made by Tomruen)

The case of the quadcopter consists of 20 regular hexagons and 12 regular pentagons with joint sides, forming a truncated icosahedron [2] (see Fig 5.11a and 5.11b). The

[2] http://en.wikipedia.org/wiki/Truncated_icosahedron

hexagons and pentagons which are built up with carbon edges and 3D-printed joints (see Fig 5.12).

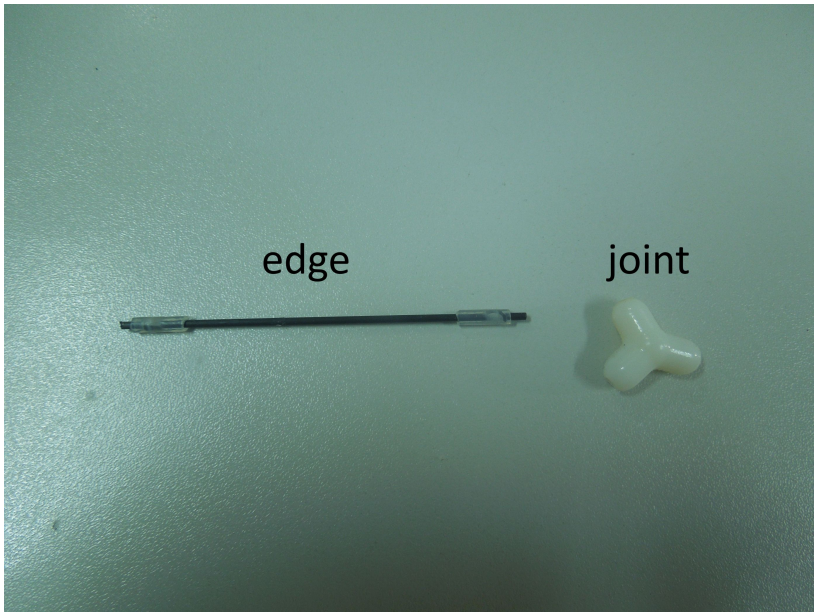


Figure 5.12: Carbon edges and 3D-printed joints

Building the case involved making several trade-offs. We must keep the weight of the case and the quadcopter as a whole in mind. The case needs to be strong enough to maintain a sphere-shaped form and not break when forces are applied, such as a player pushing it, when falling on the ground or crashing into another object.

In our first attempt we used carbon edges of 3mm thick, the case remained completely sphere-shaped, however when applying forces some joints broke immediately because all the forces were redirected to the joints. We do not want the case to break as it takes a lot of time and effort to build, therefore the edges should be more flexible to absorb a part of the force applied to the case. Therefore we reduced the thickness of the carbon edges to 1.5mm. We attached a piece of rubber at both ends of the edges which fits into the joints which helps to absorb the forces applied to the case. The holes in the case need to be small enough to prevent the user from touching the propellers, however not too small to influence the air flow of the quadcopter because this disturbs it's movements.

There are several components to the quadcopter, such as propellers, a battery, the microcontroller, the receiver, and the frame on which the components are attached (see Fig 5.13). The receiver is in our setup an Arduino which receives data with an XBee Series 1 module. The battery has a capacity of 3700mAh, contains 3 Cells at 11.1V and weighs 264g. It is connected to the speed controllers, motors and the CC3d with the discharge plug while the balance plug is connected to a custom printout. The custom printout allows the Arduino to read the remaining ampere on the battery in order for the system to use this information in the calculation of the quadcopter commands.

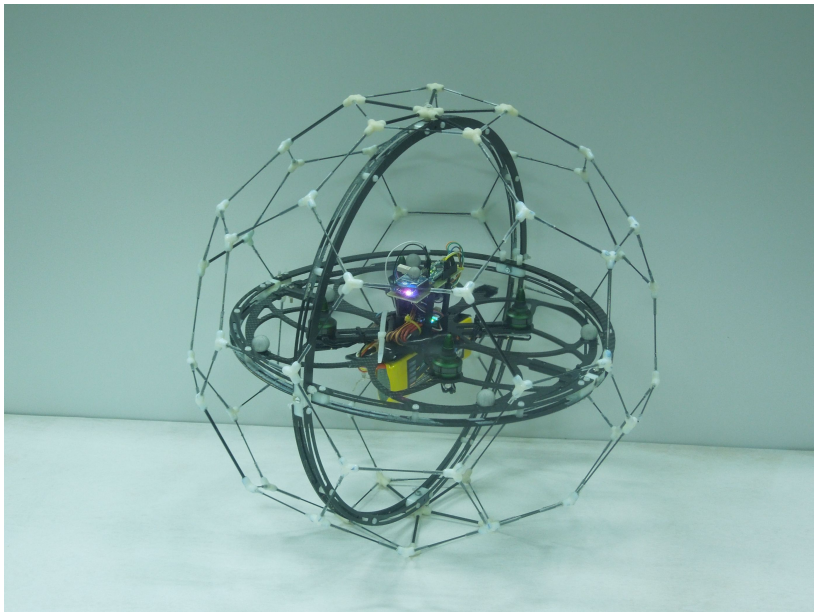
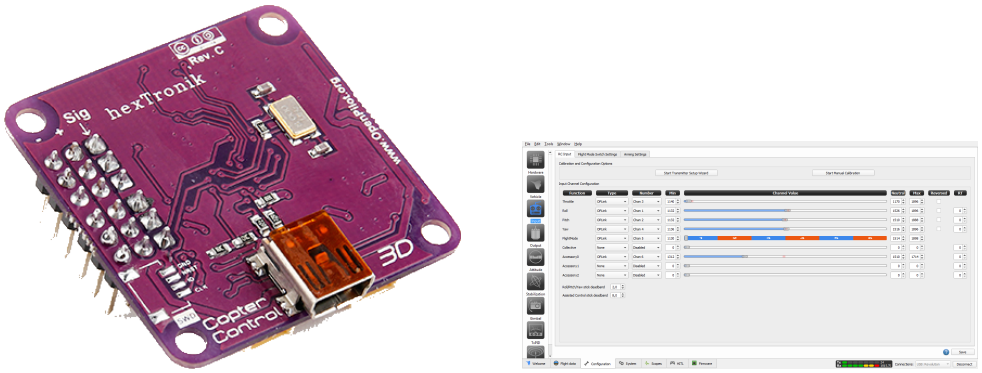


Figure 5.13: Custom built quadcopter

The microcontroller is the CC3D (see Fig 5.14a), which is an all-in-one stabilization hardware that runs the OpenPilot firmware and contains a gyroscope in order to stabilise the quadcopter. It converts quadcopter commands to commands for each of the four propellers. The CC3D is configured using the OpenPilot Ground Control Station software [3] (see Fig 5.14b).



(a) CC3D **(b)** OpenPilot Ground Control System

Figure 5.14: Quadcopter configuration

5.4 Code framework

Our code framework written in C++ is based on a game object component system as described in Game Programming Gems 6 [5, Chapter 4.6]. This system makes it easy to implement a new functionality because it increases the extendibility of the system.

The game object component system consists of game objects which are the basic things that exist in the system. Each game object has a unique name, a position, an orientation and a collection of game object components. Each game object component has a type and provides a specific functionality and implementation. There are several

[3] https://www.openpilot.org/product/openpilot-gcs/

types of game object components, some handle the physics while others handle rendering. For example if we want a game object to be subject to gravity or other physics such as collisions, we add a physics game object component to it. The physics game object component contains properties of the object, such as the dimensions, mass or inertia. Another option is to add a rendering game object component to the game object in order to display it inside a certain scene.

Only one type of game object component is allowed in each game object. For example if two physics game object components could be added to the same game object each of them would alter the position of the game object resulting in incorrect behaviour.

5.4.1 Setting up the system

Setting up the system includes two main parts, setting up the playing field and defining the teams.

The code framework allows players to define the position of secondary sports attributes as well as the dimensions of the playing field. Each ball sport requires different attributes therefore the calibration is different for each of them. For example in order to play basketball the system needs the dimensions of the playing field as well as the positions of the basketball hoops. Volleyball on the other hand requires the location and height of the volleyball net in addition to the playing field dimensions. The system asks the players to define each of the required positions. To define a position in the environment players hold a marker on a location in the environment and press a button to confirm the position.

The system allows the players to determine which player belongs to which team and to calibrate the visible players with the press of a button.

5.4.2 Physics engine

Our system uses a physics engine, Bullet [4], which is a software library that simulates physics systems, such as rigid body, soft body and fluid dynamics. Rigid body dynamics incorporates the movement and collision detection of non-deformable bodies, such as walls, under the action of external forces. Soft bodies can change their form to a certain extent thus the relative distance of two points on the object is not fixed, examples include springs or grass. Fluids can change their form completely, unlike soft bodies. Bullet is a real time physics engine, thus it uses simplified calculations and decreased accuracy to compute the physics in time for the system to respond in a timely manner.

We recreate the virtual scene in Bullet, this allows us to simulate the movement and collisions of objects based on calculations of our system. This scene contains objects, such as the floor, wall planes and players. Each object has certain attributes, such as position, orientation, size, mass, restitution, inertia and friction. Based on these attributes the physics engine determines the result of collisions between objects. When we throw an object in the scene it bounces off the walls and floor (see Fig 5.15 and 5.16).

The physics engine is sped up in order to acquire the result of certain actions faster. For example we determine the trajectory of an object which we throw in the real world before it touches the ground.

The physics engine is integrated in the system by Raf Ramakers.

[4] http://bulletphysics.org

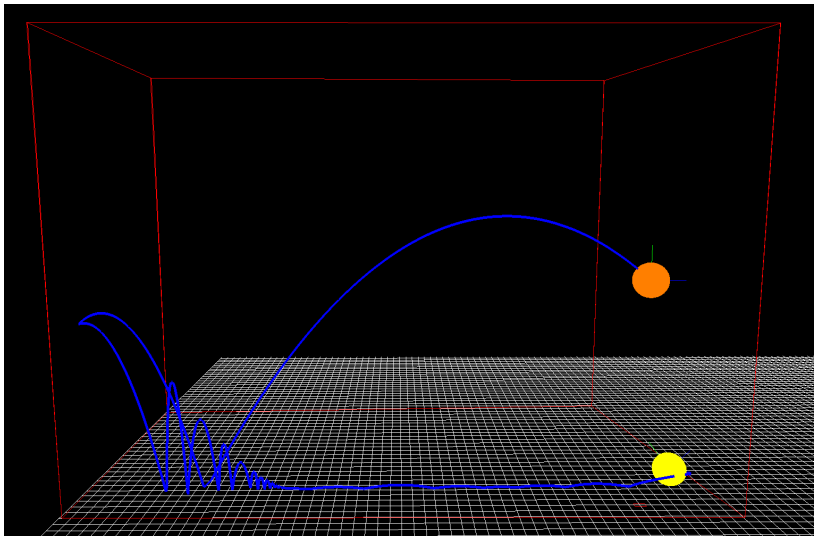
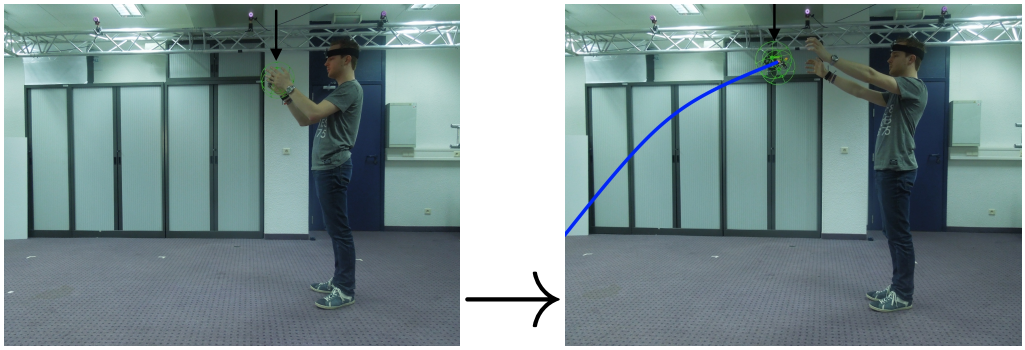
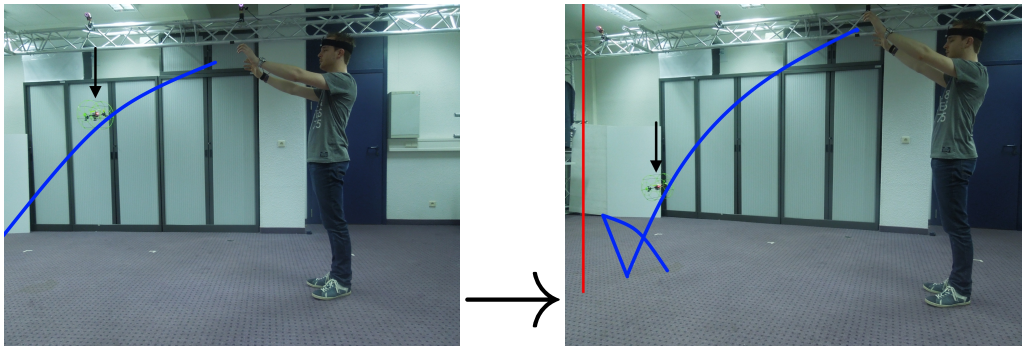


Figure 5.15: Throwing a ball in the physics engine, the orange ball indicates the start position and the yellow ball is the end position.



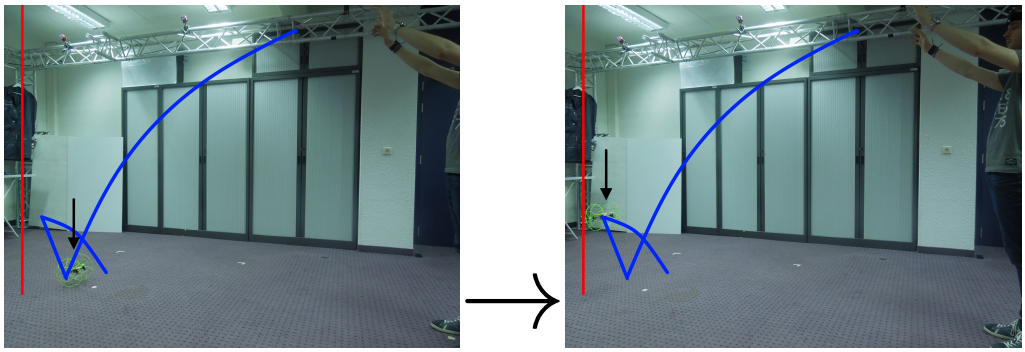
(a) The player holds the ball.

(b) The trajectory is predicted.



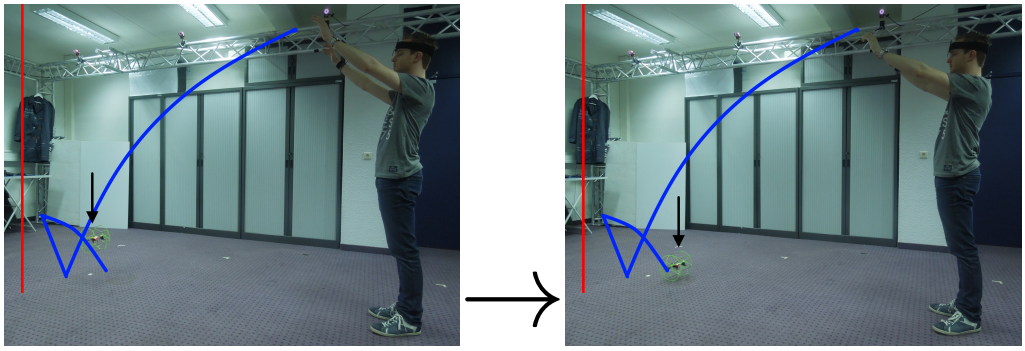
(c) The ball is following the predicted trajectory.

(d) The ball is still following the predicted trajectory.



(e) The ball hits the floor.

(f) The ball hits the virtual wall.



(g) The ball bounces off the virtual wall.

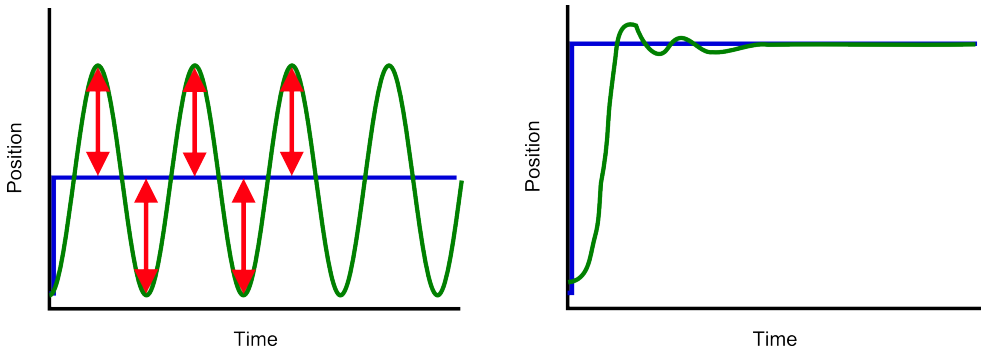
(h) The ball hits the floor a second time.

Figure 5.16: The actuated ball bounces off virtual walls and the floor in the real world.

5.4.3 Navigating the actuated ball

The system is able to move the actuated ball to a certain point in space, the destination. The direction to move the actuated ball to the destination is determined based on the current position and orientation. The current position and orientation of the actuated ball are provided by the tracking system. The commands for the actuated ball are calculated every 20 milliseconds, thus every 20 milliseconds the movement of the actuated ball is measured and corrected if necessary.

The movement from the current position to the destination has to happen without overshooting and undershooting to control the actuated ball efficiently and provide a good user experience. Overshooting happens if the actuated ball moves towards the destination with a constant speed and only starts to slow down or correct once it has passed the destination (see Fig 5.17a). The actuated ball starts to oscillate around the destination, however it will not stabilise. The desired result is displayed in Fig 5.17b



(a) Overshooting is indicated in red.

(b) The actuated ball does not overshoot nor undershoot significantly

Figure 5.17: Actuated ball PID behaviour.
The desired height is indicated in blue and the actual height over time in green.

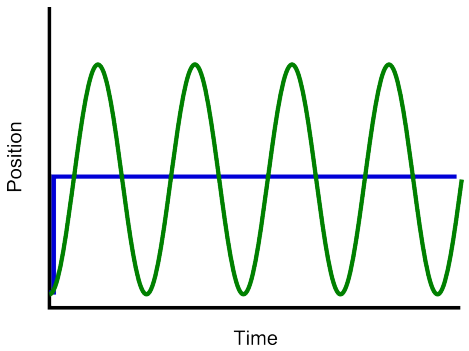
To avoid overshooting and undershooting we need to control the speed with which the actuated ball moves based on the distance the actuated ball has to fly. To control the speed with which the actuated ball moves we use a control loop mechanism, a PID controller [5], which is one of the most commonly used control loop mechanisms. A PID controller is a generic algorithm, thus some variables need to be determined individually for each process. The speed of the actuated ball is determined by the throttle and the angle it is flying, thus the speed of the actuated ball is determined by throttle, pitch and roll.

Figure 5.18: PID sliders

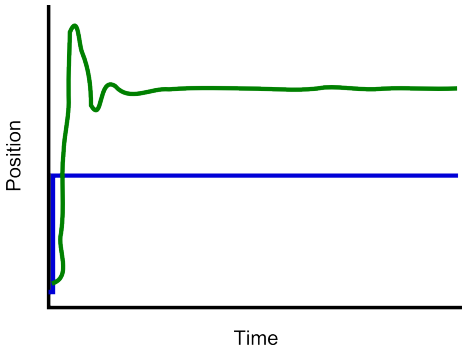
[5] http://www.embedded.com/design/prototyping-and-development/4211211/PID-without-a-PhD

A PID controller is a control loop mechanism that minimizes an error by manipulating a variable that influences the error. In our case the error is the distance to the destination and the variable is the current speed of the actuated ball. The PID controller tries to avoid overshooting and undershooting by using the current error, *P-value*, in combination with an accumulation of the past errors, *I-value*, and a prediction of the future error, *D-value*. The weighted sum of these three values, P, I and D is used to adjust the process, however not all three errors have the same weight. We determined the relative weights by hand in a manual tuning process [6] (see Fig 5.18). We tuned the PIDs seperately, first the PID for the throttle and afterwards the PIDs for both the pitch and roll.

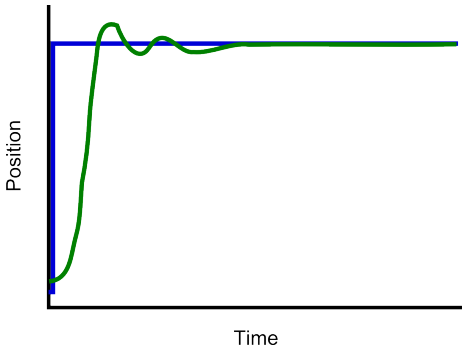
To display the effect of the P-, I- and D-values we give an example of the manual tuning process of the height PID controller. In a manual tuning process, we start with all the P-, I- and D-values at zero. We first determine the P-value by increasing it until the system oscillates, thus no stabilisation occurs (see Fig 5.19a). Once we determined the P-value, we start to increase the D-value until the system stabilises, however not necessarily on the correct height (see Fig 5.19b). Increasing the I-value shifts the height at which the actuated ball stabilises to the correct height (see Fig 5.19c).



(a) Increasing the P-value leads to oscillation.



(b) Increasing the D-value ensures the system stabilises, however not on the correct height.



(c) Increasing the I-value shifts the stabilisation height to the correct height.

Figure 5.19: Manual PID height tuning

Since we are able to navigate the actuated ball to a position the next logical step is to traverse a trajectory. A trajectory is a list of positions which the actuated ball has to pass through. The speed of the trajectory can be constant along the whole trajectory or differ from step to step. If the speed differs from step to step, each position on the trajectory also has a corresponding timestamp that indicates the time since the start of the trajectory when the actuated ball should pass through this position.

[6] http://en.wikipedia.org/wiki/PID_controller#Manual_tuning

5.4.4 Game logic

In our general-purpose system it is possible to implement many different direct contact ball sport concepts. The game logic implements the rules of the game and how the ball reacts to the actions of the players, for example how and when the trajectory is adjusted. In the current implementation, we limited the game logic to how the ball reacts to the actions of a single player.

For example, the game logic in a game of basketball manipulates the trajectory of the ball only when the player makes an sufficient effort to reach the basketball hoop. This is determined by the proximity of the prediction of the trajectory of the ball. If the player makes sufficient effort the trajectory is adjusted and the ball goes through the hoop. In the other case the ball follows the normal predicted trajectory.

5.4.5 Predicting the trajectory of the actuated ball

The first goal is to make the actuated ball act like a normal ball. In order to do this we identified the two states of the ball, either it is held by the player or not held by the player. If the ball is held by the player the system does not take any action, thus the actuated ball is idle. Only once the ball is released, for example dropped or thrown by the player, the system takes action. If the actuated ball is dropped it bounces on the floor or when it is thrown it follows the normal trajectory.

To detect whether the player holds or releases the actuated ball the system calculates the distance between the wrist of the player and the actuated ball. The player has a marker attached to his wrist in order for the system to determine the position of the wrist. If distance between the position of the wrist and the actuated ball is small enough the system detects that the player holds the ball. Once the distance between the wrist and the actuated ball becomes too big, therefore the system detects that the player released the ball.

When a player releases the actuated ball the system has to anticipate as fast as possible. To anticipate on the results of the actions of the player, we predict the trajectory of the ball in the physics engine by simulating the flight of the ball as a result of the actions of the player. The physics engine determines the trajectory of the object based on the gravity and collisions with other objects and fluids in the physics engine as previously mentioned in section 5.4.2. Based on this trajectory the game logic determines whether it adjusts the trajectory of the ball. If the game logic does not adjust the trajectory of the ball it follows the trajectory predicted by the physics engine. The physics engine predicts the flight of the actuated ball by simulating the actions of the player.

In order to simulate the action of the player in the physics engine the system needs the position and the velocity when the player released the ball. The velocity of the actuated ball is calculated based on the first ten positions of the trajectory (see Fig 5.20 trajectory A). The time between two position updates is 8.3 ms, thus 83 ms after the ball leaves the hands of the player the trajectory is predicted by the physics engine. The physics engine predicts the trajectory of the actuated ball by throwing a sphere-shaped object in the same position and with the same velocity as the player (see Fig 5.20 trajectory B). The sphere-shaped object in the physics engine has several properties such as size, mass, inertia, friction and restitution. The system applies these properties depending on the type of ball the game uses, for example a basketball has a different size and mass than a volleyball.

The physics engine runs at ten times the normal pace in order to decrease the time needed to predict the trajectory, otherwise the time to traverse the path in the physics engine is the same as in the real world and the system is unable to adjust the trajectory. However, the more we speed up the physics engine the less fluent the predicted trajectory is since it contains less points, however the trajectory itself is still correct. here we

make a trade-off between the speed of the prediction and the number of points in the trajectory.

The total time required to predict the trajectory of the actuated ball is approximately 155 ms which is divided in two parts. The first 83 ms are required to calculate the velocity of the actuated ball, based on the first ten samples as described above. The last 72 ms are used to simulate the actions of the player, for example throwing the actuated ball, inside the physics engine. These timings are averages over several runs and are influenced by how the user throws the ball, in which direction, at which speed and how far from the destination it is thrown.

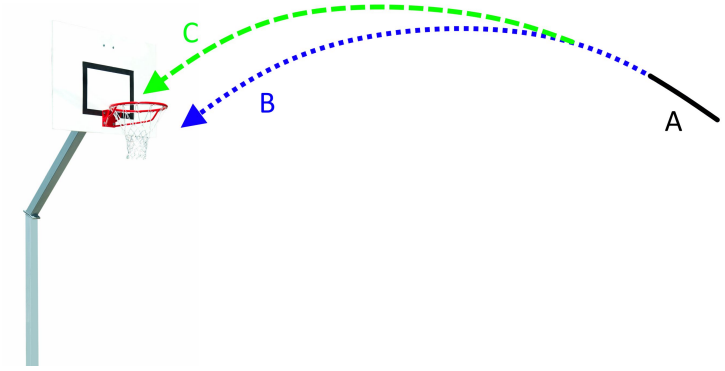


Figure 5.20: A: The part of the trajectory of the actuated ball in the real world we use to determine it's velocity .
B: The predicted trajectory of the actuated ball
C: The adjusted trajectory

5.4.6 Adjusting the trajectory of the actuated ball

After we predicted the trajectory the game logic decides whether to adjust the trajectory of the actuated ball or not. For example, in a game of basketball the game logic determines how close the ball will approach the basketball hoop, if this distance is close enough the game logic adjusts the trajectory. Adjusting the trajectory is done by calculating a parabola from the current position of the actuated ball and the destination. In the previous example the destination is the basketball hoop.

One possible solution to calculate the points of a parabola would be by using a quadratic function:

$$f(x) = ax^2 + bx + c \tag{5.1}$$

However this solution does not account for collisions with virtual objects, such as walls. Thus instead of calculating the points of the parabola with a quadratic function, the system throws a ball in the physics engine. The physics engine determines the trajectory while accounting for unrealistic experiences, such as a change of gravity or a collision with a virtual object.

The parabola starts at the current position of the actuated ball and passes through the destination. We calculate the minimum velocity that is needed to throw the ball from the first point to the second. Once the system determined the adjusted trajectory of the actuated ball by throwing a ball in the physics engine, it makes the actuated ball follow the adjusted trajectory (see Fig 5.21 and 5.22).

The total time required to determine the adjusted trajectory of the actuated ball is approximately 200 ms which is divided in three parts. The first two are described in the previous section and take approximately 155 ms. They respectively are used to calculate the velocity of the actuated ball and to predict the trajectory of the actuated ball. The

last part is used to determine the adjusted trajectory from the current position of the actuated ball to the destination and takes 45 ms.

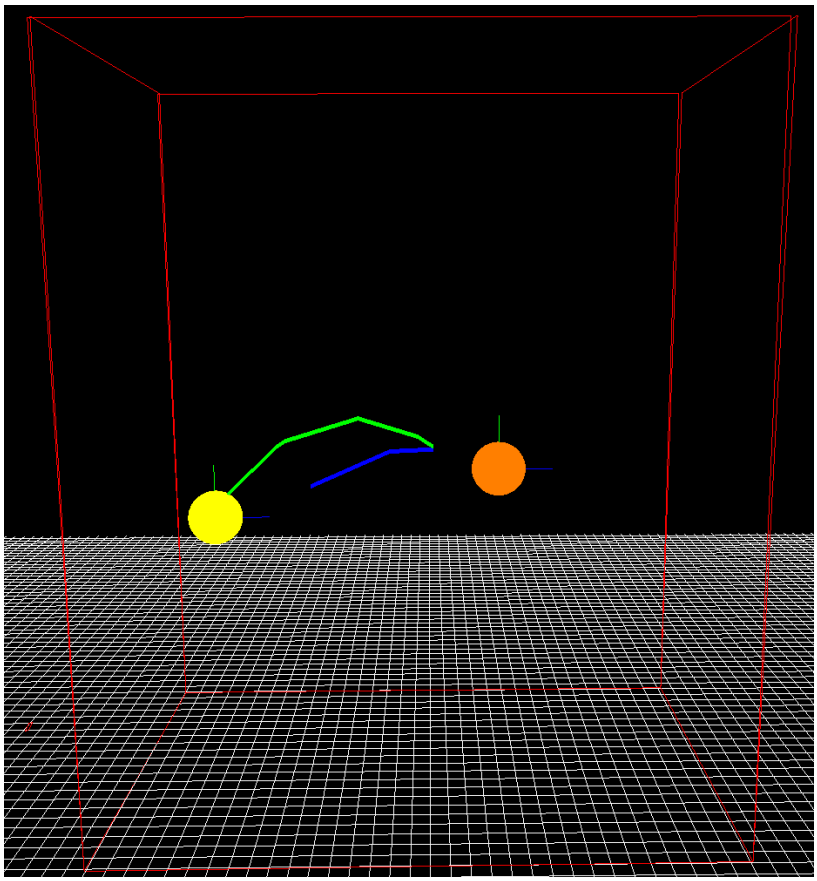
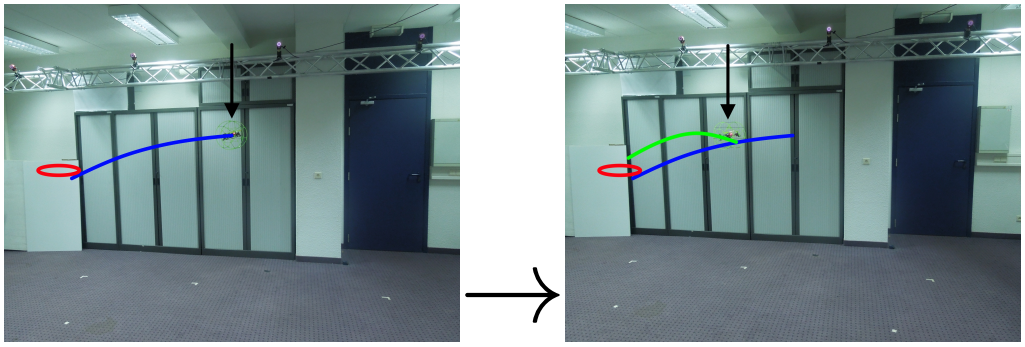
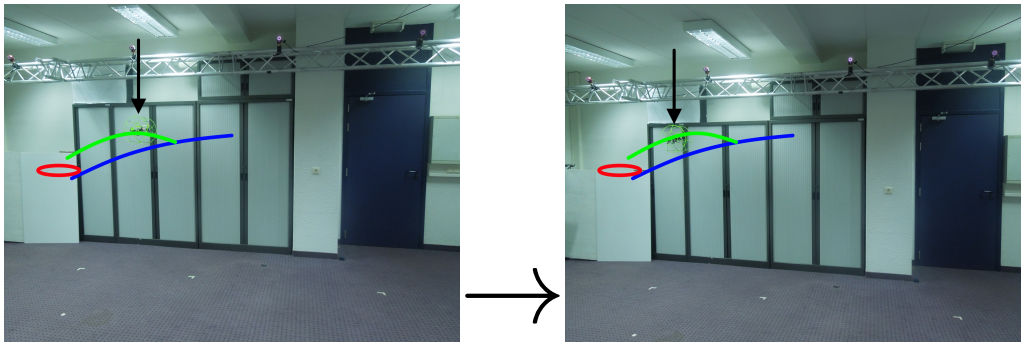


Figure 5.21: We want to reach the destination, the yellow ball, by throwing the orange ball. The predicted trajectory is displayed in blue, which closely misses the destination. The adjusted path is a parabola and is displayed in green.



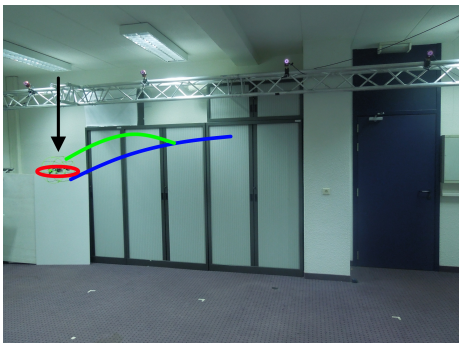
(a) The ball follows the predicted trajectory displayed in blue. The adjusted trajectory will be determined because the predicted trajectory is close enough to the basketball ring

(b) The ball starts to follow the adjusted trajectory, displayed in green.



(c) The ball continues to follow the adjusted trajectory.

(d) The ball continues to follow the adjusted trajectory.



(e) The ball reaches the basketball ring.

Figure 5.22: The trajectory of the actuated ball is adjusted in the real world. The adjustment happens based on the decision of the game logic.

When the ball passes through the destination, the actuated ball follows the complete trajectory. This is only the case until a player holds the ball, for example by picking it up from the ground or intercepting it. At that moment the system stops intervening in the gameplay until the player releases the ball at which point the system will predict the trajectory again.

5.4.7 Bouncing off virtual objects

All secondary sports objects are virtual, thus invisible or projected in the real world. They are defined at the beginning of the game as explained in section 5.4.1. The physics engine predicts the trajectory of an object while taking collisions with other virtual

objects into account. For example if we throw an object in the real world we simulate this throw in the physics engine in order to predict the trajectory. The predicted trajectory bounces off virtual walls (see Fig 5.23 and 5.24).

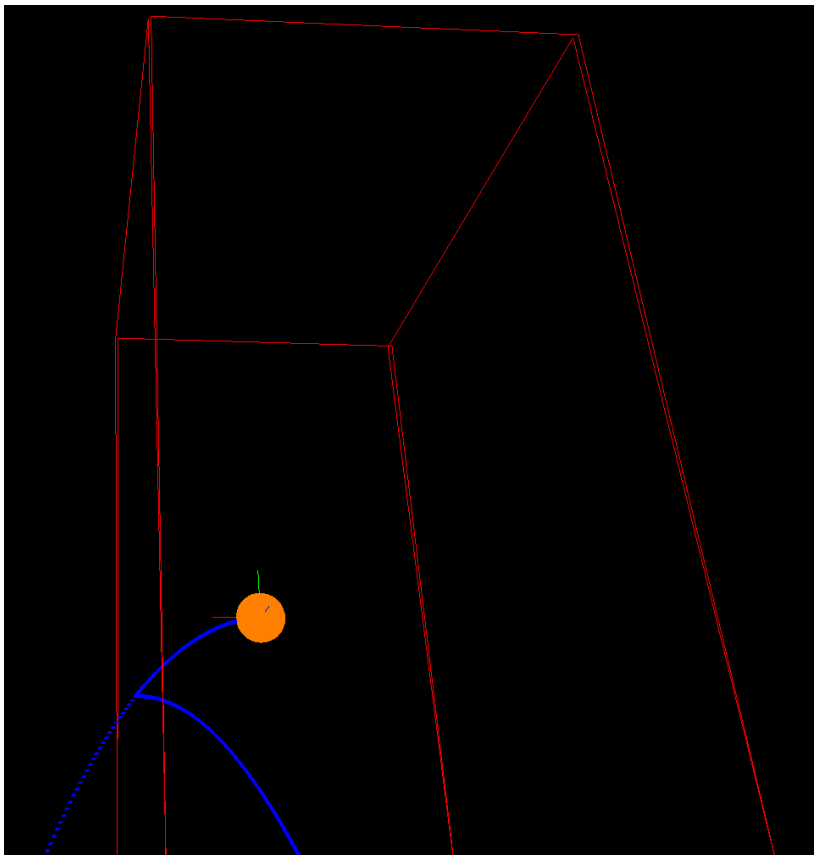
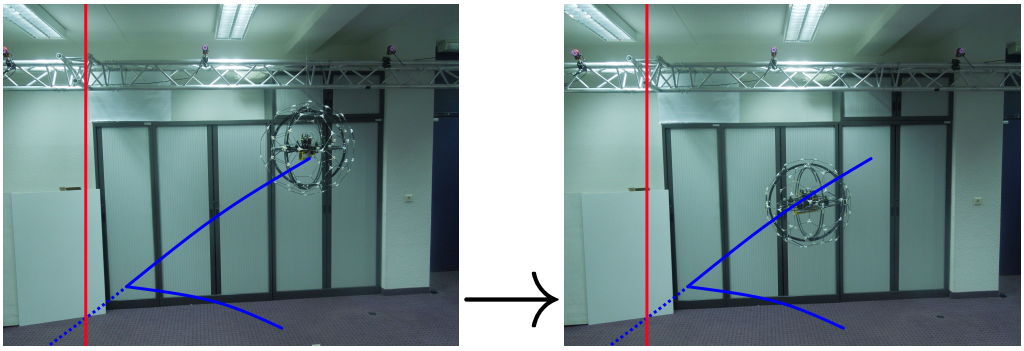
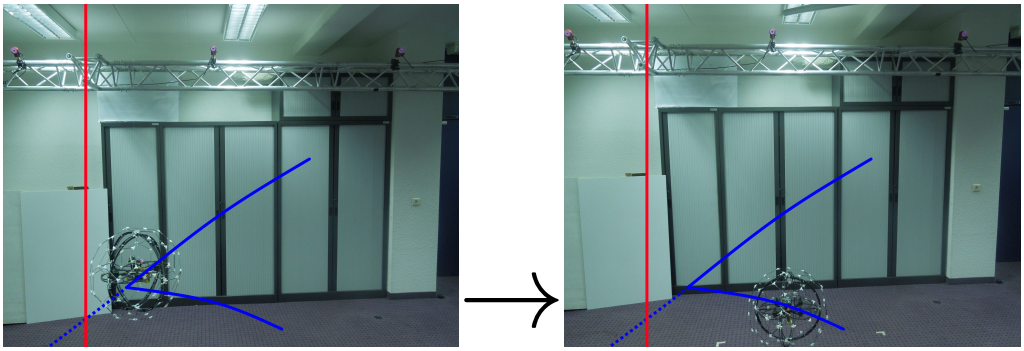


Figure 5.23: Throwing the actuated ball in the real world from the location of the orange ball in the direction of the virtual wall. The trajectory in the real world is displayed with the dotted blue line. The predicted trajectory of the object in the physics engine is displayed in a full blue line which bounces off the virtual wall. The actuated ball follows the predicted trajectory, thus the full blue line.



(a) The start position of the ball.

(b) The ball follows the predicted trajectory.



(c) The ball hits the virtual wall.

(d) The ball bounces off the virtual wall and changed direction.

(e) The ball hits the floor.

Figure 5.24: The actuated ball bouncing off a virtual wall

5.5 Conclusion

In this chapter we provided an overview of the general-purpose ball sports platform and illustrated the implementation of the current system. The system architecture consists of the tracking system, an actuated ball and the code framework. We described the interactions between the different parts of the system and how this results in the adjustment of the trajectory of the actuated ball.

Chapter 6

Discussion

6.1 Introduction

In this thesis we combined the tangibility of traditional ball sports with the flexibility of virtual games so that multiple games can co-exist in a single general-purpose ball sports platform. This removes the need to reconfigure the entire environment and move secondary sports attributes, such as basketball loops or nets, around. In addition to this we introduced non-realistic experiences, such as changing the gravity of the actuated ball, to our platform by mounting a quadcopter inside the ball in order to precisely change the trajectory. In this chapter we discuss the current state of the system.

6.2 Tracking system

The tracking system is an optical tracking system based on infared light and passive retro-reflective markers which provides sub millimetre absolute positioning. It functions properly once all the cameras are each attached to an immobile object, pointed in the right direction and the system is calibrated. However the setup is rather bulky, requires a certain amount of space and is hard to move from one place to another.

6.3 Actuated ball

We use an actuated ball to introduce unrealistic experiences in the real world. In the current implementation we used two different actuated balls with distinct capabilities.

6.3.1 Toy actuated ball

The toy actuated ball is very light, only 84 gram, however the throttle required to make it hover is high, about 80%. The case is very flexible, but very strong since it can withstand a lot of crashes. While tuning the actuated ball crashes are inevitable, therefore a strong actuated ball is required. It is a simple actuated ball which does not produce too much noise. The system navigates it accurately and makes it traverse a trajectory.

The communication with the actuated ball was good in the beginning of the experiment, however after some time the communication quality degraded. Once the actuated ball becomes unusable due to the degraded communication we try to reconnect multiple

times. It takes a lot of time for the actuated ball to start communicating again properly, which is time-consuming. The more we use the actuated ball, the less reliable the communication becomes, thus we conclude that the communication is not very reliable. As a result the actuated ball would not function properly all the time, in the end it does not function any more. Therefore we were unable to continue our experiments with this actuated ball.

The actuated ball can only be rotated around one axis without destabilising it. A weakness of this actuated ball is that when big forces are applied to it, it takes time to stabilise. It does not matter whether the direction of these forces is perpendicular to the axis or not. The reason for this is that the actuated ball is too weak in order to compensate for the external forces because it already needs 80% throttle to overcome gravity. Therefore it is not possible to throw it and expect a fast stabilisation and it is less useful for our purposes.

6.3.2 Custom actuated ball

Since the toy actuated ball is not suitable for our purpose, we built a custom actuated ball ourselves. Building the custom actuated ball required making several trade-offs regarding the case and the overall weight. The actuated ball needs to withstand big forces, for example when it crashes. It must be able to rotate in every direction while it stays stable. The actuated ball needs to be safe, players are not allowed to touch the propellers. The total weight must be as light as possible in order to react fast.

The case is flexible in an attempt to withstand big forces, however in reality it still breaks rather fast because the 3D-printed joints break too easily. During the construction of the custom actuated ball we reduced the thickness of the edges of the case from 3mm to 1.5mm in an attempt to make them more flexible and prevent the 3D-printed joints from breaking, however this attempt is unsuccessful. In order to keep the ball stable while rotating it in any direction we used two gimbals which functions properly. The holes in the case are too big, because players are still able to touch the propellers.

The required throttle to make the actuated ball hover is about 90% of the maximum value because of the total weight of the ball, which is around 900 gram. The higher the throttle is that the actuated ball needs to hover, the faster the battery drains which results in an unstable flying experience because small corrections are harder to make.

The power that goes to the propellers is influenced by the remaining capacity of the battery, thus to hover at the same height the required throttle increases as the remaining battery capacity decreases. In the current implementation, thus by using the toy actuated ball, this is covered by the increasing I-value in the throttle PID, however this process is slow and lowers the performance of the actuated ball. A better approach is to search for a relation between the remaining battery capacity and the required throttle adjustment and integrate this in the system.

We did not succeed in tuning the actuated ball, in the limited time, because of three main reasons. The first reason is the high required throttle which makes complicates tuning the quadcopter because small adjustments are harder to make. The second reason is the fact that the case breaks often, till ten times a day, and repairing it takes a lot of time. The third reason is the fact that the remaining battery capacity heavily influences the behaviour of the actuated ball. Our solution to the last problem included finding a relation between the remaining battery capacity and the required throttle so this can be used in the system by adjusting the throttle. However in order to do this the actuated ball needs to be able to stay in the air for a while, therefore our attempt is unsuccessful.

The actuated ball itself is too big to represent an average bal, therefore we need to make a smaller and lighter actuated ball with enough power and comparable flight

time. It is difficult to observe the audio feedback because this is prevented by the noise disturbance of the actuated ball, therefore the actuated ball has to be more quiet.

6.4 Code framework

The current implementation features a general purpose ball sports platform which combines the advantages of digital ball sports with the advantages of their real world counterparts. The strengths of digital ball sport games include unrealistic experiences, but also the fact that they are not reliant on physical attributes. Real world ball sports do feature physical attributes which allows players to use the skill they developed with balls over a lifetime. However all attributes are physical which is a hassle to manually position them. Therefore the system renders all secondary sports attributes, such as basketball hoops and volleyball nets, virtual and only the primary sports attribute, the ball, is present in the real world. Virtual attributes can be invisible to the user or projected into the real world. Players define the locations of the secondary sports attributes and the dimensions of the playing field by holding a marker on a certain position and pressing a button. The system is able to detect the location of the players and the hand of each player. The trajectory of objects, in this case the actuated ball, is predicted which the system uses in order to react accordingly.

The system intervenes in the gameplay by adjusting the trajectory of the actuated ball, thus changing reality. The system detects whether the player made sufficient effort to achieving a certain goal, such as scoring a point in basketball game, by predicting the trajectory of the ball. The system is only adjusts the trajectory of the actuated ball if the player made sufficient effort to achieve the goal. The initial goal of this thesis is to create a general-purpose ball sports platform in which we create multiple ball sports concepts. Due to time constraints we were unable to create a ball sports concept. Since we did not create a ball sports concept we focused on one non-realistic experience, the adjustment of the trajectory.

The system is able to navigate the toy actuated ball accurately, however not the custom built actuated ball. The custom actuated ball took longer to construct than anticipated and it took more effort to tune it. Tuning any actuated ball is time-consuming, however tuning the custom actuated ball took extra effort because the case or other parts break fast. We were unable to implement a full game due to the fact that the toy actuated ball is not suitable for this purpose.

6.5 Conclusion

The goal of this thesis is to create an general-purpose ball sports platform with an actuated ball. The most important requirement is the actuated ball, the creation of which is very challenging because of all the aspects that need to be taken into account. The long time needed to create and tune the actuated ball influenced the creation of the code framework. The current actuated balls do not meet all the requirements. In our current implementation we used our first actuated ball because this is the only one our system can navigate safely.

Chapter 7

Future work

7.1 Introduction

In this thesis we explored the opportunities of a general-purpose ball sports platform. We combined the advantages of digital ball sports with the advantages of their real world counterparts by using actuated balls. By introducing unrealistic experiences into the real world we are able to change reality. However the current implementation is not finished, as we discussed in the previous chapter. In this chapter, we elaborate on the next steps to be taken based on the conclusions made in the previous chapter.

7.2 Tracking system

The tracking system provides sub milimetre precision position and rotation information and functions properly, however it has some drawbacks. The system is expensive and takes some time to set up and calibrate. Users are required to wear markers on their head and wrist and markers have to be attached to the actuated ball in order for the system to identify them. The setup itself is rather bulky because all the cameras need to be immobile in order to not disturb the calibration. Because the system is optical, it is prone to occlusions. Adding more cameras helps to fix this partially however it still remains a problem.

A possible solution is to replace the tracking system with a setup with multiple kinects. This would solve the price problem and remove the need for markers on each user. The system is easier to set up. However this solution is far from perfect, it does not solve occlusion problems nor is it certain that the accuracy will suffice.

7.3 Actuated ball

As we concluded in the discussion the toy actuated ball is of the right size, not too heavy, does not break and it is accurately controllable by the system. On the other hand, it has problems with a low flight time, requires high throttle to hover, is unable to handle big forces, and cannot rotate around all axes. The custom built actuated ball is able to rotate around all axes while staying stable and has no problems with flight time. However it is too big and too heavy, it requires high throttle to hover, it does break easily and is currently not accurately controllable by the system.

First of all, we need to create an actuated ball with enough power to operate properly, thus under 70% throttle in order to hover. Secondly the actuated ball has to be more

sturdy, it has to be able to withstand big forces. Thirdly the actuated ball needs to stabilise quickly.

The actuated ball needs to be a lighter version of the custom actuated ball with comparable power and flight time. The weight of the actuated ball is predominantly determined by the size and weight of the battery. In order to create an actuated ball that flies longer, a bigger and often heavier battery is required. The added weight from the bigger battery mostly decreases the flight time because the required throttle the actuated ball needs to hover increases. The size of the actuated ball does not matter that much however the current custom actuated ball is very big. A smaller ball would make it easier for players to interact with it.

In order to make the actuated ball more sturdy, constructing the case completely out of carbon might be a viable option. In order to still be able to open the case it could be constructed as two halves spheres which can be attached and detached easily.

The actuated ball needs to be more quiet, in the current iteration it is not possible to observe the audio feedback because the sound of the actuated ball is louder. Currently audio feedback is provided by the speakers which are often located on one position outside of the playing field. However in most ball sports the majority of the audio comes from the collisions of the ball with other objects, for example when players interact with it. Therefore it is logical to provide audio feedback from inside the actuated ball. This slightly increases the weight of the actuated ball, but makes it easier to provide localised audio.

When we have a working prototype it would be interesting to study if actuated ball sports are more engaging and more motivational than the non actuated ball sports.

7.4 Code framework

In this thesis we did not implement a full game concept due to time constraints and technical problems with the custom actuated ball, however in the future we will implement several ball sports, for example basketball, volleyball, korfball, pass the ball, waterpolo, bowling, etc.

In our current implementation the predicted and adjusted trajectory of the ball is a set of separate points due to the speedup of the physics engine. They are connected by straight lines in order to make a trajectory for the actuated ball to follow, however this trajectory is not smooth. In a next iteration we will apply a Bézier curve to the trajectory to smoothen it.

The system only tracks the positional and angular displacement of the actuated ball and not for example the torque, therefore it is unable to detect all the ball skills the player uses. For example it does not detect when a player gives a spin to the ball, which greatly influences the trajectory of the ball when there is a collision with another object.

Currently the players wear a marker on their wrist in order for the system to determine when the players hold and release the ball. It is possible to determine this without a marker based on the values of the PID of the quadcopter, however due to time constraints this was not implemented in this iteration.

Currently the implementation does not take the remaining battery capacity into account. However the remaining battery capacity influences the power that is supplied to the propellers and therefore the actuated ball flies lower with less capacity which has a negative effect on the performance. In a next iteration we should increase the throttle when the remaining battery capacity decreases.

The unrealistic experiences are virtual, thus currently invisible for the user. Although sound feedback is useful, visual output is more clear for the user. In order to preserve

the social interaction and the virtuality of the unrealistic experiences projecting the secondary sports objects could be a solution, for example using the RoomAlive Toolkit [1].

7.5 Conclusion

This chapter provides solutions to the weaknesses of the current implementation. We propose to replace the optical tracking system with a kinect based tracking system. We summarise the recommended changes to the actuated ball, most notably changes regarding lower throttle and increased strength of the case. The first addition to the code framework is implementing a full game. The second important addition Other additions include measuring the torque applied to the actuated ball in order to take the ball control skills of the players into account.

[1] http://research.microsoft.com/en-us/projects/roomalivetoolkit/

Chapter 8

Conclusion

The work in this thesis is based on combining the best of digital ball sports and their real world counterparts in a general-purpose ball sports platform. We create actuated ball sports by introducing non-realistic aspects in the real world by using actuated balls.

After an investigation of existing exertion games, we concluded that some of them are specifically made with one concept in mind. In order to create more general-purpose systems, non-realistic aspects are made virtual which allows to implement different concepts with the same system, however without physicality. To regain this physicality we use actuated balls which is controlled from a distance, therefore also preserving the non-realistic aspects. The physical ball addresses the dexterity of the human body and the skills humans developed with these balls over a lifetime.

The system engages in the gameplay by adjusting the trajectory of the ball towards a goal if the user made enough effort into reaching that goal. We focused on predicting and adjusting the trajectory of the ball and making the ball bounce off virtual walls, however there are many more possible ways of engaging in the gameplay which will be developed in the near future.

In this thesis, we did not implement a complete game concept. We succeeded in adjusting the trajectory of the ball, however the actuated ball used in the implementation is unable to handle the external forces fast enough when thrown. Given the proper actuated ball it is possible to create the presented system. Creating a suitable actuated ball is hard, because it has to take many aspects into account, however we are convinced this will be possible in the near future.

Chapter 9

Nederlandstalige samenvatting (Dutch summary)

Introductie

In deze thesis gaan we onderzoeken of het mogelijk is om een universeel platform te maken waarin we verschillende sporten kunnen spelen met ballen die we vanop afstand kunnen besturen. De ontwikkeling en het gebruik van computerspelen heeft de laatste jaren een enorme evolutie doorgemaakt. Vroeger konden we enkel instructies geven via het toetsenbord of andere randapparatuur zoals een controller. Systemen zoals Wii en Kinect brachten hier verandering in en stimuleerden de speler om hun hele lichaam te gebruiken. Hoewel deze systemen de spelers aanzetten tot beweging moeten de spelers nog steeds gefocust zijn op het scherm, waardoor ze de gelaatsuitdrukkingen en de lichaamstaal van andere medespelers niet kunnen zien. Hierdoor is er minder sociale omgang in vergelijking met echte sporten. Deze sociale omgang is nochtans belangrijk voor de ontwikkeling van mensen omdat we hier veel levenservaring uit kunnen putten. Anderzijds laten computerspelen toe om allerlei virtuele werelden en situaties te creëren die het spelen extra aantrekkelijk kunnen maken. Bepaalde spelers kunnen bevoordeeld of benadeeld worden omdat ze bijvoorbeeld minder of net meer spelervaring hebben. Dit is niet mogelijk in de echte wereld. Balsporten in de echte wereld brengen fysieke attributen met zich mee, zoals ballen, basketbalringen en volleybalnetten. Hierdoor is er een beperking in het aantal spelen dat we kunnen spelen omdat we vaak al deze attributen niet beschikbaar hebben. We gaan dus onderzoeken of we een universeel platform kunnen maken in de echte wereld waarbij we verschillende balspelen kunnen beoefenen zonder dat we deze attributen nodig hebben. Dit gaan we realiseren door deze attributen onzichtbaar te maken in de echte wereld, maar ze enkel virtueel te creëren. Daarnaast gaan we zorgen dat de spelers volledig sociaal contact hebben met elkaar en niet enkel met elkaar communiceren via een scherm. Anderzijds willen we ervoor zorgen dat we zoveel mogelijk voordelen van de digitale balsporten behouden zoals het bevoordelen of benadelen van spelers.

Bestaande systemen

Er zijn al een aantal systemen die spelers aanzetten om hun hele lichaam te gebruiken. Deze systemen hebben we onderzocht. Sommige van deze systemen zijn specifiek gemaakt met slechts één doel waardoor het moeilijk is om deze te gebruiken voor meerdere doeleinden. Deze systemen zijn vaak gebaseerd op één sport waarbij ook de attributen fysiek aanwezig moeten zijn. Sommige andere systemen kunnen universeel worden gebruikt,

dus voor meerdere doeleinden. Een manier om een universeel systeem te maken is door virtualisatie. Hiermee bedoelen we dat het spel in de echte wereld wordt gespeeld, maar dat bepaalde delen van het spel zich in de virtuele wereld bevinden. Een voordeel hierbij is dat het mogelijk is om meerdere sporten te beoefenen zonder de opstelling aan te passen. Een tweede voordeel is om onrealistische virtuele aspecten, zoals omgekeerde zwaartekracht, in het systeem te integreren. Deze oplossing heeft echter ook enkele nadelen. Er is vaak minder feedback, dikwijls beperkt tot visuele feedback, wat de sociale interactie niet ten goede komt. Doordat een deel van de systemen virtueel zijn, is er geen fysieke interactie waardoor spelers hun balvaardigheden niet kunnen gebruiken. Om deze fysieke interactie te behouden zonder de onrealistische virtuele aspecten te verliezen, gebruiken wij in ons systeem een bestuurbare bal in de echte wereld binnen een universeel platform.

Concept

Ons concept bestaat uit een universeel platform zodat we heel snel kunnen wisselen van balsport. We zorgen ervoor dat het fysieke contact met de bal blijft bestaan, omdat het succes bij een balspel voor een deel bestaat uit de effecten die we kunnen uitoefenen op de bal. Hiermee bedoelen we bijvoorbeeld de spin die we aan de bal geven en die bepalend kan zijn of de bal in de basketbalring gaat of dat alle kegels vallen bij bowling. We willen gebruik kunnen maken van onrealistische aspecten, zoals het wijzigen van de zwaartekracht of het bevoordelen van een speler of team, maar ook zorgen dat de fysieke en sociale interactie tussen de spelers blijft bestaan. We kunnen dus de realiteit veranderen. Hiervoor maken we software waarmee we de bal kunnen besturen. Dit kan nuttig zijn om een bepaald evenwicht in het spel te bewaren of om het spel aantrekkelijker te maken. Een voorbeeld om een bepaald evenwicht te bewaren, is wanneer kinderen tegen hun meer ervaren ouders spelen, we ervoor kunnen zorgen dat de kinderen sneller een punt kunnen maken door de bal een andere koers te geven indien ze voldoende moeite doen. Een voorbeeld om het spel aantrekkelijker te maken, is dat we ervoor kunnen zorgen dat de bal op een bepaald moment, of tijdens het volledige spel, beweegt alsof we het spel op de maan zouden spelen. Of dat de beweging boven een bepaalde hoogte anders verloopt dan onder deze hoogte, zoals bij waterpolo de weerstand boven of onder water ook verschillend is. We creëren dus een platform omdat we de voordelen en nadelen van digitale balsporten en hun tegenhangers in de echte wereld willen combineren.

Systeem

In ons platform kunnen we drie belangrijke delen onderscheiden. Het eerste is een tracking systeem dat op ieder ogenblik de exacte positie van de bal en de spelers kan bepalen (zie figuur 1.1). Dit tracking systeem moet zeer snel en betrouwbaar reageren. Als voorbeeld kunnen we geven dat wanneer we een voorwerp van een meter hoogte laten vallen, zonder er ook maar enige kracht op uit te oefenen, dit voorwerp op de grond valt in minder dan een halve seconde. Om deze reden hebben we een systeem nodig dat minimum elke 10 milliseconden ons de nieuwe positie en rotatie doorgeeft van alle elementen op het speelveld. Het tweede is een bestuurbare bal die zo snel mogelijk in alle richtingen kan bewegen. De bal moet voldoende lang kunnen blijven vliegen en moet dus licht zijn om de batterij te sparen. Hij moet ook voldoende krachtig zijn om bij een wijziging, zoals het inbrengen van een onrealistisch spelelement, ogenblikkelijk te kunnen reageren. Maar hij moet ook voldoende sterk zijn om een val, of een slag van een speler op de bal, op te vangen. Als derde en laatste belangrijk element in ons concept hebben we de software. Deze moet ervoor zorgen dat we de bal op eender welk moment kunnen controleren en dus naar een bepaalde bestemming kunnen sturen en zijn snelheid kunnen aanpassen. Aangezien we via ons tracking systeem op eender welk

moment weten waar de bal zich bevindt kunnen we snel ingrijpen. Ook moet de software in staat zijn de regels van het spel te controleren. Via deze software moeten we ook snel en eenvoudig nieuwe balsporten kunnen maken en dus ook de grootte van het speelveld en de locatie van de diverse spelattributen kunnen aanpassen. Deze drie delen van ons platform moeten dus op een zeer accurate manier samenwerken.

Implementatie

Het tracking systeem dat wij hebben gebruikt is een Optitrack tracking systeem. Dit is een optisch systeem met twaalf camera's dat ervoor zorgt dat we een exacte plaats en oriëntatie van de voorwerpen hebben met een nauwkeurigheid van minder dan 1 millimeter. Dit systeem geeft ons 120 beelden per seconde, dus elke 8.3 milliseconde krijgen we een nieuwe positiebepaling. Het systeem werkt via de camera's die infrarood licht uitzenden. Dit licht wordt opgevangen door reflecterende markers. De infraroodcamera's vangen dit licht ook terug op en zo weet het systeem op ieder moment waar alle voorwerpen zich bevinden. Dit infraroodlicht is onzichtbaar voor het menselijk oog. Het systeem vereist dat er minstens drie camera's zijn die een bepaalde reflecterende marker zien. De reflecterende markers op een object moeten een uniek patroon vormen zodat het systeem zeker weet welk voorwerp zich waar bevindt. Deze reflecterende markers zijn immers bevestigd op zowel de bestuurbare bal als op het voorhoofd en de pols van de spelers. De bestuurbare bal bestaat uit een quadcopter, dit is een soort drone, die aan de buitenkant door een omhulsel wordt beschermd. Een quadcopter wordt normaal bestuurd door een afstandsbediening, maar wij sturen de quadcopter aan via ons systeem. Ons systeem is dus in staat om de quadcopter in alle richtingen te bewegen.

We hebben twee bestuurbare ballen gebruikt. Onze eerste was een speelgoed bestuurbare bal (zie figuur 5.9). Deze weegt maar 84 gram en kan maximum 6 minuten in de lucht blijven. Deze speelgoed bestuurbare bal heeft echter enkele minpunten. Hij kan niet rond elke as draaien. Dit heeft als nadeel dat, als hij te veel gedraaid wordt in de verkeerde richting, hij niet snel genoeg hersteld en dus te lang blijft oscilleren. Deze bestuurbare bal heeft onvoldoende krachtterugkoppeling. Hierdoor verliest hij snel hoogte wanneer er kracht op wordt uitgeoefend of duurt het lang om terug op zijn geplande traject te komen. De reden hiervoor ligt ongetwijfeld in het feit dat de speelgoed bestuurbare bal 80% van zijn capaciteit nodig heeft om omhoog te gaan. Hij heeft dus maar een beperkte reserve om externe krachten te overwinnen die op hem inwerken. We stellen wel vast dat het besturen via ons systeem veel beter lukt dan een manuele besturing via de controller. Om de eerder vermelde problemen op te lossen, maakten we een op maat gemaakte bestuurbare bal. Deze weegt ongeveer 900 gram en kan 12 minuten vliegen. Deze kan in alle richtingen roteren omdat hij bestaat uit een quadcopter die gemonteerd is in twee op elkaar ronddraaiende ringen. Hierdoor kan deze vlotter in alle richtingen navigeren. Eromheen is een balvormig omhulsel gemaakt om de bal te beschermen tegen externe schokken. Bijna de volledige constructie is gemaakt van carbon, dit zou ervoor moeten zorgen dat het geheel sterk en licht is. De tussenstukjes voor de constructie van het balvormig omhulsel zijn geprint met een 3D printer. De communicatie tussen de computer en de bestuurbare ballen gebeurt draadloos. Onze software is geschreven in C++ en is gebaseerd op het game object component systeem, zodat we ons systeem in de toekomst gemakkelijk kunnen uitbreiden. Elk voorwerp heeft een bepaalde naam, plaats en oriëntatie.

Als we willen starten met een spel moeten we enkel in het systeem het speelveld opzetten en de teams bepalen. Het speelveld en de fysieke attributen, zoals de basketbalring of de hoogte van het volleybalnet, bepalen we door een reflecterende marker op deze plaats te houden en te bevestigen in het systeem. Hetzelfde gebeurt voor de bepaling van de spelers door iedere speler een unieke reflecterende marker te geven en deze dan op het speelveld te koppelen aan zijn plaats op het speelveld door een druk op de knop in ons systeem. Zo kunnen we ook de teams aan elkaar koppelen. Als het

spel bezig is gaat ons systeem het spel continu monitoren. De spelregels zitten geprogrammeerd in het systeem. Als het systeem beslist om in te grijpen in het normale spelpatroon om bijvoorbeeld een speler te helpen om een bal door de ring te gooien, dan gebeurt dit als volgt. Het systeem bemerkt dat een speler de bal gooit doordat de afstand tussen de reflecterende marker op de bal en de reflecterende marker op de pols plots sterk toeneemt. Het systeem berekent het normale traject dat de bal gaat afleggen. Op basis hiervan zal de software, op basis van eerder ingebrachte criteria, beslissen of er wordt ingegrepen in het spel. Als dit niet gebeurt zal het systeem ervoor zorgen dat de bal dit berekende normale traject zal volgen. Als het systeem wel ingrijpt in het spel, zal bijvoorbeeld het traject van de bal worden aangepast (zie figuur 5.22). Nadat de bal gegooid is, wordt na tien nieuwe positiebepalingen beslist of het traject van de bal zal worden aangepast. Dit is dus na 83 milliseconden omdat we elke 8.3 milliseconden een nieuwe positiebepaling binnen krijgen. Dan heeft ons systeem gemiddeld nog 72 milliseconden nodig om de berekening te maken welke aanpassing vereist is en dan duurt het gemiddeld nog 45 milliseconden voordat het traject daadwerkelijk is aangepast. Dus 200 milliseconden nadat de bal is gegooid is het traject volledig aangepast. Het is belangrijk dat de eventuele aanpassing van het traject zeer geleidelijk gebeurt. Er is voorzien dat we dit kunnen aanpassen in het systeem. Dit gebeurt door ervoor te zorgen dat de bal snel naar zijn nieuwe positie gaat zonder echter te ver te gaan en zonder te oscilleren rond de nieuwe positie. In dit hoofdstuk hebben we besproken hoe we ons systeem hebben uitgewerkt en welke toepassingen we al hebben getest.

Bespreking

Ons opzet is om een universeel spelplatform te maken waarbij we gebruik maken van een bestuurbare bal en waarbij we secundaire spelattributen virtueel bepalen. Het tracking systeem werkt perfect tijdens onze testen. Het is echter een systeem dat weinig mobiliteit toelaat gezien het vele werk dat nodig is om het op te zetten.

Zoals al aangehaald hebben we twee bestuurbare ballen gebruikt. De eerste speelgoed bestuurbare bal was licht, had een beperkte vliegtijd, was niet snel beschadigd bij een val, maar heeft het nadeel dat hij lang oscilleert wanneer er een te grote kracht op uitgeoefend word. Met deze bal lukt het wel om een aanpassing van het traject te doen, maar door het oscilleren, duurt het telkens heel lang voordat de bal stabiliseert. In het begin was de communicatie tussen onze computer en deze bal heel goed maar na verloop van tijd werd dit minder en minder. Om onverklaarbare reden viel deze communicatie meer en meer weg en kunnen we deze bal nu niet meer bedienen. De tweede, op maat gemaakte bestuurbare bal is veel zwaarder, heeft een langere vliegtijd, maar is zeer snel beschadigd bij een val. De 3D geprinte tussenstukjes breken zeer snel en het vraagt telkens zeer veel tijd om dit te herstellen. De gaten in het balvormig omhulsel zijn ook te groot zodat het nog steeds mogelijk is om met de handen de propellers te raken. Deze op maat gemaakte bestuurbare bal kan in alle richtingen draaien. Bij deze bal moeten we zelfs 90% van de capaciteit benutten om het toestel in de lucht te krijgen. Bij deze bal stellen we ook vast dat er een omgekeerd verband bestaat tussen de benodigde capaciteit om te vliegen en de resterende batterijcapaciteit. Naarmate er meer en meer ontlading is moeten we de capaciteit nog opdrijven om te vliegen. We zijn er niet in geslaagd om deze op maat gemaakte bestuurbare bal af te stellen en op een veilige manier te laten vliegen. Hiervoor zijn er drie redenen. Door de hoge benodigde capaciteit is het veel moeilijker om het toestel af te stellen, om correcties uit te voeren. De tweede reden is omdat het balvormig omhulsel zeer dikwijls brak, tot wel 10 keer per dag, en het veel tijd kost om dit te repareren. De derde reden is dat de resterende batterijcapaciteit de reactie van de bal sterk beïnvloedt.

Onze software werkt goed, we zijn erin geslaagd om een traject aan te passen, de plaats van de basketbalring te bepalen en een bal tegen de virtuele muur te laten botsen. We zijn er niet in geslaagd om een volledig spel te ontwikkelen, dit voornamelijk omwille

van de verschillende problemen met de bestuurbare ballen en de tijd die we hebben besteed om deze ballen klaar te maken voor ons project. Om deze reden hebben we ook geen andere onrealistische aspecten zoals een aanpassing van de zwaartekracht kunnen implementeren.

Toekomstig werk

Zoals ook al uit het vorige hoofdstuk blijkt, zijn er nog zeer veel mogelijkheden om het systeem te verfijnen en uit te breiden. Het tracking systeem zou nog verbeterd kunnen worden door meer camera's te plaatsen, omdat we nu soms merken dat door de aanwezigheid van andere objecten op het speelveld er geen drie resterende reflecterende markers zichtbaar zijn, wat een vertraging geeft in het systeem. Aan het ontwerp van de ideale bestuurbare bal is nog het meeste werk. Door de vaak tegenstrijdige belangen is het moeilijk om hier een goed evenwicht in te vinden. Hij moet licht en toch sterk zijn met een grote batterijcapaciteit voor een lange vliegtijd. Hij zou ook kleiner moeten zijn als de huidige op maat gemaakte bal, met kleinere openingen zodat er veiliger mee kan gevlogen worden. De vereiste capaciteit om op te stijgen moet lager zijn om zo eenvoudiger correcties te kunnen uitvoeren, zodat het afstellen gemakkelijker wordt en de bal sneller stabiliseert. Zoals al gemeld hebben we geen volledig spel kunnen implementeren, dus dit is een eerste opdracht. Daarna kunnen we nog diverse andere balspelen introduceren. We kunnen het systeem nog verfijnen zodat de overgang van het normale traject naar het aangepaste traject vloeiender verloopt. Ook kunnen we nog rekening gaan houden met effecten die de speler op de bal uitoefent, zoals spin, om meer natuurgetrouwe resultaten te verkrijgen. In een volgende versie zouden we ook moeten implementeren dat het systeem rekening houdt met de resterende batterijcapaciteit zodanig dat deze geen invloed heeft op de reactie van de bal. Er zijn dus nog vele mogelijkheden om ons project te vervolledigen.

Besluit

In deze thesis willen we het beste van twee werelden combineren, de voordelen van digitale balsporten en de voordelen van gewone balsporten, door gebruik te maken van bestuurbare ballen. Dit om de fysische aspecten van ballen te behouden en toch gebruik te kunnen maken van de onrealistische aspecten die digitale balsporten kunnen bieden. Daarenboven willen we dit realiseren binnen een universeel platform waarbinnen we verschillende balsporten kunnen spelen. In eerste instantie hebben we ons toegelegd op het aanpassen van het traject en het botsen van de bal tegen de virtuele wand. In de nabije toekomst zijn er nog diverse andere toepassingen mogelijk. Doordat we veel tijd hebben moeten besteden om een werkende bestuurbare bal te bekomen, zijn we er niet in geslaagd om een volledig spel te ontwikkelen. Als we er in de toekomst in slagen om een adequaat werkende bestuurbare bal te maken, wat niet eenvoudig is gezien de moeilijk verenigbare vereisten, zijn we ervan overtuigd dat onze studie zal leiden tot de realisatie van het voorgestelde systeem.

Bibliography

[1] Ancient egyptian sport. URL http://www.sis.gov.eg/En/Templates/Articles/tmpArticles.aspx?ArtID=1694#.VIa7oHW9_CI.

[2] *Evaluating User Experience in Games*, chapter 11: Evaluating Exertion Games. Springer London, 2010.

[3] Patrick Baudisch, Henning Pohl, Stefanie Reinicke, Emilia Wittmers, Patrick Lühne, Marius Knaust, Sven Köhler, Patrick Schmidt, and Christian Holz. Imaginary reality gaming: Ball games without a ball. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 405–410, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2268-3. doi: 10.1145/2501988.2502012. URL http://doi.acm.org/10.1145/2501988.2502012.

[4] Hrvoje Benko, Andrew D. Wilson, and Federico Zannier. Dyadic projected spatial augmented reality. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 645–655, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3069-5. doi: 10.1145/2642918.2647402. URL http://doi.acm.org/10.1145/2642918.2647402.

[5] M. Dickheiser. *Game Programming Gems 6*. GAME DEVELOPMENT SERIES. Charles River Media, 2006. ISBN 9781584504504. URL https://books.google.be/books?id=lPZQAAAAMAAJ.

[6] Eberhard Graether and Florian Mueller. Joggobot: A flying robot as jogging companion. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 1063–1066, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1016-1. doi: 10.1145/2212776.2212386. URL http://doi.acm.org/10.1145/2212776.2212386.

[7] Takashi Ichikawa and Takuya Nojima. Development of the motion-controllable ball. In *Adjunct Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 425–426, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0462-7. doi: 10.1145/1866218.1866253. URL http://doi.acm.org/10.1145/1866218.1866253.

[8] Robert J.K. Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. Reality-based interaction: A framework for post-wimp interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 201–210, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-011-1. doi: 10.1145/1357054.1357089. URL http://doi.acm.org/10.1145/1357054.1357089.

[9] Florian 'Floyd' Mueller and Martin Gibbs. A table tennis game for three players. In *Proceedings of the 18th Australia Conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments*, OZCHI '06, pages 321–324, New York, NY, USA, 2006. ACM. ISBN 1-59593-545-2. doi: 10.1145/1228175.1228234. URL http://doi.acm.org/10.1145/1228175.1228234.

[10] Florian 'Floyd' Mueller and Matthew Muirhead. Jogging with a quadcopter. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 2023–2032, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702472. URL http://doi.acm.org/10.1145/2702123.2702472.

[11] Florian 'Floyd' Mueller, Shannon O'Brien, and Alex Thorogood. Jogging over a distance: Supporting a "jogging together" experience although being apart. In *CHI '07 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '07, pages 2579–2584, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-642-4. doi: 10.1145/1240866.1241045. URL http://doi.acm.org/10.1145/1240866.1241045.

[12] Florian 'Floyd' Mueller, Stefan Agamanolis, Martin R. Gibbs, and Frank Vetere. Remote impact: Shadowboxing over a distance. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '08, pages 2291–2296, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-012-8. doi: 10.1145/1358628.1358672. URL http://doi.acm.org/10.1145/1358628.1358672.

[13] Florian 'Floyd' Mueller, Darren Edge, Frank Vetere, Martin R. Gibbs, Stefan Agamanolis, Bert Bongers, and Jennifer G. Sheridan. Designing sports: A framework for exertion games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2651–2660, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0228-9. doi: 10.1145/1978942.1979330. URL http://doi.acm.org/10.1145/1978942.1979330.

[14] Kei Nitta, Keita Higuchi, and Jun Rekimoto. Hoverball: Augmented sports with a flying ball. In *Proceedings of the 5th Augmented Human International Conference*, AH '14, pages 13:1–13:4, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2761-9. doi: 10.1145/2582051.2582064. URL http://doi.acm.org/10.1145/2582051.2582064.

[15] Kei Nitta, Toshiki Sato, Hideki Koike, and Takuya Nojima. Photoelasticball: A touch detectable ball using photoelasticity. In *Proceedings of the 5th Augmented Human International Conference*, AH '14, pages 16:1–16:4, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2761-9. doi: 10.1145/2582051.2582067. URL http://doi.acm.org/10.1145/2582051.2582067.

[16] Tomoya Ohta, Shumpei Yamakawa, Takashi Ichikawa, and Takuya Nojima. Tama: Development of trajectory changeable ball for future entertainment. In *Proceedings of the 5th Augmented Human International Conference*, AH '14, pages 50:1–50:8, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2761-9. doi: 10.1145/2582051.2582101. URL http://doi.acm.org/10.1145/2582051.2582101.

[17] Wayne Piekarski and Bruce Thomas. Arquake: The outdoor augmented reality gaming system. *Commun. ACM*, 45(1):36–38, January 2002. ISSN 0001-0782. doi: 10.1145/502269.502291. URL http://doi.acm.org/10.1145/502269.502291.

[18] Emma Witkowski. Running from zombies. In *Proceedings of The 9th Australasian Conference on Interactive Entertainment: Matters of Life and Death*, IE '13, pages 1:1–1:8, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2254-6. doi: 10.1145/2513002.2513573. URL http://doi.acm.org/10.1145/2513002.2513573.