DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Inspector Dance - Social Dancing and Deduction Game based on IMU Gesture Recognition

Nina Cordes

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics: Games Engineering

Inspector Dance - Social Dancing and Deduction Game based on IMU Gesture Recognition

Inspector Dance - Soziales Tanz- und Deduktionsspiel basierend auf IMU Gestenerkennung

Author:	Nina Cordes
Supervisor:	Prof. Gudrun Klinker, Ph.D.
Advisor:	M.Sc. Sandro Weber
Submission Date:	15.09.21

I confirm that this bachelor's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Garching, 15.09.21 Nina Cordes

Acknowledgments

Thank you to Linda for pointing me in the right direction in the beginning and of course thank you to Sandro for always making the meetings fun and for letting me do what I had envisioned for this thesis.

Also a big thanks to Jonas for keeping me more or less sane and for fixing the LaTeX-Editor whenever it decided it did enough work for today.

Abstract

People are sitting too much and exercising too little in modern society. This is mostly because of time and motivation problems. To combat the problem of motivation, an engaging and interesting to play game that involves exercising with friends was developed. It combines elements of dancing and social deduction games. For the recognition of dance movements, a smartphone is used as a controller with IMU sensors and a computer displays the visuals. Points are distributed via an euclidean distance metric, after evaluating multiple different recognition methods for their advantages and disadvantages. To visualize the movements, the direction is determined separately from the points. Moreover, the IMU data received from the smartphone is analyzed to identify issues and propose solutions.

The development of a lobby system for the Ubi-interact framework is documented and explained in detail.

It was found that the direction of movements is hard to determine based on the IMU measurements, whereas the euclidean distance metric works quite well. Further tests would have to be carried out after improving the visualization of the players' movements, but an initial test indicates that the concept is fun for the players and has potential.

Contents

1 Introduction

Although sitting is well established in our society, it has a big problem: people are not made for spending most of their day in a sitting position. This can lead to physical or mental health consequences [1]. Nowadays, a lot of young people spend their time working on their computers and in their free time they like to play games on that same computer. This is not generally a bad thing since games have been proven to enhance cognitive abilities and reaction time as well as mental health [2]. But it is not good for their bodies to be sitting all day long. Movement is important for staying healthy and fit. But sports can be boring and if you can not meet with others, or do not have enough free time, it is even more discouraging.

This is why we want to make a game with some connection to sports where you can get some exercise into your day without even really noticing it. The problem is, not everybody has a platform specially made for sports games at home. What we can use instead are cameras of day to day electronics or smartphones because they have IMU sensors with which they can be tracked similarly to controllers made for this purpose.

We could use the smartphone-controller to track any kind of movement of the arm/hand but we decided to do dancing because it is fun and is not only good physical exercise but also stimulates rhythm and simply makes people happy as well as training the brain simultaneously [3]. If a form of exercise is fun, people are more likely to stick to it, change their lifestyle to a less sedentary one and get the health benefits that it brings.

Social deduction games fascinate people and are played around the world. Incorporating this concept into an exercising game promises to make the game addicting and encourage people to play it more often.

This is why we decided to make a dancing game with a smartphone and a computer which also incorporates social deduction aspects.

2 Related Work

We will look at prior work in the fields of IMU, gesture recognition and similar games to get an overview of the subjects we need to make our game.

2.1 IMU

In 2019, there was a Bachelors Thesis at TUM [4] that used the Ubi-interact framework to build a mobile tennis game. The game was played with two smartphones, one used as a racket and one in a head mounted display to project the court onto the floor. They tried multiple recognition approaches for this game and came to the conclusion that Hidden Markov Models (HMM) provided the best classification rates. They distinguished between 3 different strokes. [4].

MIT conducted a study [5]on how IMUs can be used to detect and classify movements back in 2002. They build a sensor just for this purpose and predicted that in the future, more people will have access to these kind of sensors in hand-held computers. And their prediction was accurate: nowadays, practically everyone has a smartphone and those contain IMUs amongst other sensors.

The authors also proposed a very simple gesture recognition algorithm based on how many spikes the acceleration shows. Considering an arm can not move into the same direction for an infinite time because it is only so long, it has to stop at some point. This can be used to simplify the problem and just use the number of spikes in acceleration data, which is the number of direction changes. So if there are two spikes, the movement is only in one direction. The first spike is the acceleration of getting the device to move and the second one is the stopping. If there are three spikes, they classify it as a there-and-back motion and so on. Multiple movements are then combined to get more complex gestures. [5]

2.2 Gesture Recognition

Apart from detecting movements in the air, surface gestures, a gesture that involves touching or pressing on a surface, can also be detected. This was researched in a paper from 2018 [6] where the first wrist-worn device that can classify surface gestures at multiple force levels was introduced. Before that, the sensors had to be worn on the forearm, which is uncomfortable and is therefore less likely to be done in day to day life. Wrist-worn approaches could be integrated into watches or similar devices and used to interact more naturally with other devices around us. For classification, they used a machine-learning approach that is trained at the beginning once and then re-calibrated for one minute before each test. The new data is combined with the historical data to make one great algorithm. This led to high classification rates while still being a reasonable amount of work [6].

To make recognition more accurate, other sensors can be combined with IMUs. This can be EMGs like in the last example or for example acoustic measurements [7]. In this study, a gyroscope, an accelerometer and microphones were used to determine which gesture was performed. This could be integrated into smartwatches. The microphones are used as pressure-based sensors to measure the vibrations generated by the skin due to the movement of tissues and muscles under the skin. They found that the microphones increased the accuracy by almost 7% compared to just using IMU alone. Overall, the model had an accuracy of 75% and if used by the same person, even almost 80%. These percentages could still be increased by using a neural network to maximize classification accuracy [7].

The recognition of gestures can not only be used for games, it can also be useful for interacting with devices we need in our everyday life. An example for this is [8], a wheelchair controlled by gestures, which was created by a team from IIEST in India. This provides an easier and more intuitive interaction method. They managed to get a recognition rate of 90.5% while operating the wheelchair through an obstacle course. The gesture was determined complete when the hand did not move for a period of time by using standard deviation of the last 50 samples of the accelerometer readings [8]. This shows that gesture recognition can be used for serious applications, too.

2.3 Social Deduction Games

A social deduction game is a game where the players are divided into two or more groups, typically an "evil" and a "good" group. The "good" people then need to see through the lies of the "evil" to find and eliminate them. A key characteristic is the communication during the game [9].

The first successful social deduction game was Mafia, it was created in 1986 by a Russian student. Later, "The Werewolves of Millers Hollow" followed and was received quite well. Recently, "Among Us" has been very successful as it has been played by hundreds of millions of people across the world [10].

There has been a lot of research on social deduction games, such as how they can be used for teaching [11] or what kind of data the players rely on while they are playing [12]. It was found that the players like to use what the others say and how they say it as a basis for their decisions as well as non-verbal cues [12]. This shows the emphasis on communication and social skills that is typical for a social deduction game.

2.4 Dance Games

Dancing is something humans have been enjoying for a long time, so it is only natural that digital dancing games have been developed. An example is Dance Dance Revolution which came out in 1998 and was mainly played in arcade halls around the world [13]. Later, other games followed, some relying on optical tracking, others on controllers with IMU sensors. Just Dance Now is even playable on a computer with a smartphone as a controller [14].

A lot of people are very eager to dance and they have fun doing so [3], so it is very natural that there are many games out there that revolve around moving along with the beat.

There are also a lot of other games with the core concept of following a rhythm, such as Geometry Dash [15] or Piano Tiles [16] for mobile. These games are not about dancing, but you also need to press buttons based on the rhythm of a song which is a comparable activity. It also enhances the feeling for rhythm and trains the eye-hand-coordination.

3 Game Design

In this section, the basics and mechanics of the developed game are described.

3.1 Gameplay

Similar to other social deduction games, there are two groups of players: one faker and the others are inspectors. There are two slightly different versions of the same dance. One version is shown to the faker and one is shown to everybody else. Every player tries to replicate the movements they see on the screen. At the end of the song, the dance ends and every inspector has to vote on who they think the faker is. If they are correct, they get points. The faker gets points for every inspector that was fooled by them.

But there are not only points for finding the faker, there are also points for dancing. The better the dance moves are executed, the more points. The game measures the accuracy of a move and its timing and calculates a score based on that.

The game can be played with three to eight players. To make it a true social deduction game, there needs to be free communication between the players before they vote. To achieve this, they can use an independent voice chat, such as Discord, because the game itself does not have a functionality for communicating.

3.2 Implementation

We do not want to use a camera because of space concerns (see chapter 4), so we can not display the videos of all players and use this as a reference to determine who the faker is. Instead, only the direction in which they are moving their smartphone is displayed as seen in figure 3.1.

Because of this, we need to select gestures that only move along one axis. These movements can be varied in speed and paired with different foot-movements to make the dance more challenging and interesting.

During the first test (see chapter 8) we also concluded that the game is harder than we anticipated so the movements should not be as hard and repeat themselves quite often to make the game not too challenging. We also considered that the gestures must

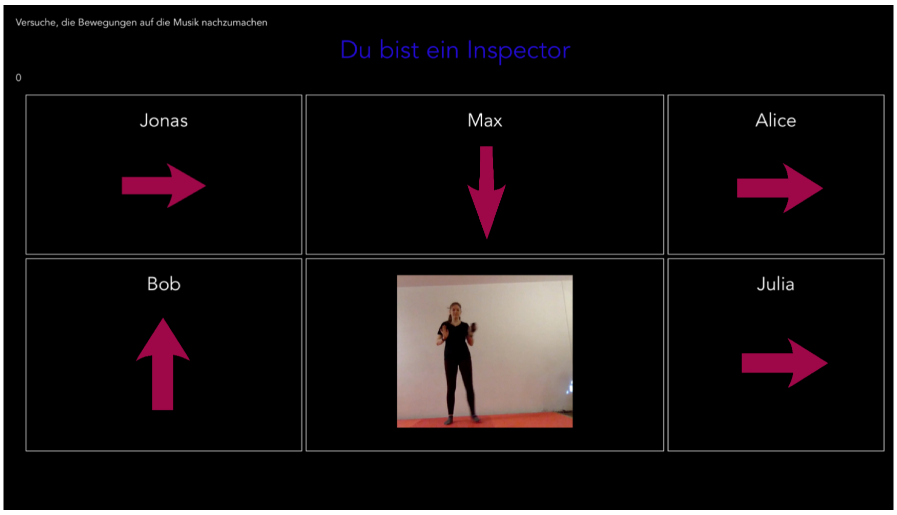


Figure 3.1: Screenshot of a typical game with five players, edited

be detectable with a sensor device in only one of the hands. So they either have to involve both arms or alternate between using the right and the left arm. The legs and rest of the body are not tracked at all so the relevant movements have to involve the arms.

3.3 A Typical Game

For a typical round of *Inspector Dance*, a player would go on the webpage and scan the QR-Code with their smartphone. Then, the smartphone needs to be placed onto a flat surface and directed toward the computer or Laptop. The player then presses the calibrate button and afterwards gives the IMU Permissions so the game is able to access IMU data on the phone.

One would then select an existing lobby to join or create their own and wait for others. Once there are enough people, the game can be started by the creator of the lobby. If someone wants to leave the lobby before that, they can do so. Once the game is started, the direction arrows of all the players are displayed in a grid with the dance that the player is supposed to do in the middle so it is easy to keep everything in sight. At the top of the screen there is a text that informs you if you are the faker or an inspector like seen in figure 3.1. The player takes the phone into their right hand and does the dance as well as they can.

When the dance is finished, all the players are automatically forwarded to the endscreen-page. There, the inspectors need to guess which one of the other players was the faker. If they guess right, they get points, if not, they lose the same amount. The faker gets points for every inspector that is wrong and loses some for every right guess. Everyone has to guess before continuing. The winner is displayed in an alert window and the players are forwarded to the desktop page where they can choose to join another lobby or quit the game by simply closing the browser.

3.4 Other, Similar Games

There are a lot of games that involve dancing. We will shortly explain the most important ones.

3.4.1 Dance Dance Revolution

In this early dancing game from 1998, the user has to press buttons according to arrows they see on the screen. These buttons are big and on the floor so they get pressed with the feet. “Dance Dance Revolution” is often found in arcade halls but when played at home, a normal controller can be used. Then the player is not really dancing except if they buy a special dancing mat to use as an controller. There is even an e-sports league and tournaments for this game.

The developers of this game later also made a game for the Kinect, “Dance Evolution” or also called “Dance Masters”. It was not received as well as Dance Dance Revolution and got some mixed reviews, for example in [13], because of the song choice and the recognition not being as good.

3.4.2 Just Dance

“Just Dance 2021” is the latest of a series of games made by Ubisoft Entertainment that works very similarly compared to our game but does not have the social deduction game aspect to it. It can be played by multiple dancers, the goal is to get as many points as possible together. Most popular platforms support it [17].

“Just Dance Now” is part of this series and can be played with a smartphone as a controller, just like we envision for our game [14]. In all other ways, it works the same as Just Dance 2021

4 Hardware

Like already mentioned, we want to develop an approach that uses hardware the user already has at home. But first, we need to take a look at all methods to find the best one.

4.1 Possible Solutions

There are multiple possibilities to record and interpret gestures from a moving person, which are now discussed.

4.1.1 Optical Tracking

The most obvious approach for gesture detection in a fixed setting is optical tracking, where a camera is used to capture all the movements of the player. Classification algorithms can then determine where the different body parts of the player are by refining and processing the picture. Afterwards, shapes are detected and a skeleton is constructed. More often than not, machine learning is used to reliably detect important features in the image, especially when dealing with problems such as self-occlusion or bad lighting. Those are, however, still problematic and decrease accuracy. The algorithms also need a lot of computation power [18].

There are also RGB-D cameras, where the D means "depth" and implicates that there is a second measurement method that creates a depth-map by using a time-of-flight camera, structured light or a similar technology. The advantage of this is that it can work in low-light environments and track at longer distances, although it requires even more filtering to get useful data [19].

Structured light approaches can also have difficulties working outside in the sunlight because it is hard to distinguish between the light sent out and the ambient lighting. Sunlight contains a broad spectrum of wavelengths and is therefore more problematic than a setup inside where the environment is more controlled [19]. Time-of-flight systems are a little more robust in outdoor conditions but if it gets too bright, the sensor gets too much light and can not function any more either [20].

4.1.2 IMU in Smartphones

IMU is an abbreviation for "internal measurement unit", a collection of sensors, most of the time consisting of an accelerometer, a gyroscope and a magnetometer [21]. Such sensors are built into all modern smartphones.

The accelerometer measures the linear acceleration, the gyroscope the angular velocity. If there is a magnetometer, it is used for calibration, making the gyroscope data more stable in order to deal with drift issues better. Nevertheless, it can be disturbed by ferromagnetic materials so it has to be used with caution [21].

By combining the data, we can get the linear acceleration and orientation, among others. While the resulting orientation is quite stable because of the extra calibration with the magnetometer, the acceleration is not. We therefore can not reliably determine the position of the sensor from the IMU data because to achieve this, we would have to integrate twice and since the data is quite noisy, this gets very unstable.

Some users drop their phones from time to time. This can damage the sensors and lead to more inaccurate measurements compared to a custom-build sensor.

The advantage of IMU sensors is that they are quite small and cheap, but the cheaper they are the more computation and filtering is necessary to get good data out of them as shown in [18].

IMU sensors have a lot of applications, for example they are often used in combination with GPS to bridge times when the GPS Satellites are not reachable because of a tunnel or something similar. It can also be used to assist indoor navigation like seen in [22].

4.1.3 Ultrasonic Tracking

We can also use another existing sensor of the smartphone: An approach called Dolphin uses the microphone and speaker of smartphones to determine the in-air gesture performed [23]. This works because of the Doppler effect, it is possible to determine the movement from how the sound was altered by the air. In this particular study, a sound with 21kHz is emitted and then registered by the microphone.

Indeed, the gesture is only tracked if the user is not more than one meter away from the device which helps prevent confusion between multiple people. In contrast to using an IMU, the user does not hold the device but it is stationary, which may be more comfortable for the player. This method has quite a high accuracy of 93% but it consumes a lot of energy [23].

4.2 Implementation: Hardware

Like we just saw, there are many possibilities to deploy our game on some kind of hardware. What we kept in mind when choosing the right technique was the accessibility to the average young person to emphasise the casual exercise aspect of the game.

For detecting the movements there are two general possibilities: use existing hardware or build a custom-designed device. Obviously the custom-design variant would have to be distributed and most likely bought additionally, so it is not a good option for our application. From all the existing controllers, we chose to use a standard smartphone because it is the most used device with the capability of recording IMU data.

We did not choose to use the camera because to do it correctly, you have to be able to see the whole body on it. However, this is hard to achieve in a small room because the player might not be able to step back far enough to be fully visible to the camera. This is why we chose a different approach.

The data for the gesture recognition will be collected by the IMU sensors in the phone, the visuals will be displayed on a computer. These two devices are found in a lot of households and are therefore easily accessible. It would be even better to only use one device but this is not possible in our case because we need a big enough display so the players can see each others' actions.

For displaying the game, we could have chosen a head-mounted display, however this is not standard in every household so we chose to stay with a regular desktop computer as a display for our game. This is also sufficient for this game and the head mounted display would not add much to the game but could impose a safety risk.

The ultrasonic approach was not chosen because one meter distance to the device is not enough for a dancing game to work properly. When standing this close, there would be the risk of hitting the device.

4.3 Deployment

Originally, we wanted to let the classification algorithms run only on the server and not on the clients devices. However, there were some difficulties with Ubi-interact and the algorithms are thus executed on the client's PC. The smartphone does the gathering of the data and then sends it to the Ubi-interact server which just passes the data on to the PC. There it is processed and the visuals are updated accordingly. An overview of the devices and their relationships can be seen in figure 4.1. The dashed lines represent logical relationships that are actually realized via the server. Only along the solid lines is there actually data traffic. The connection between the PC and the smartphone is

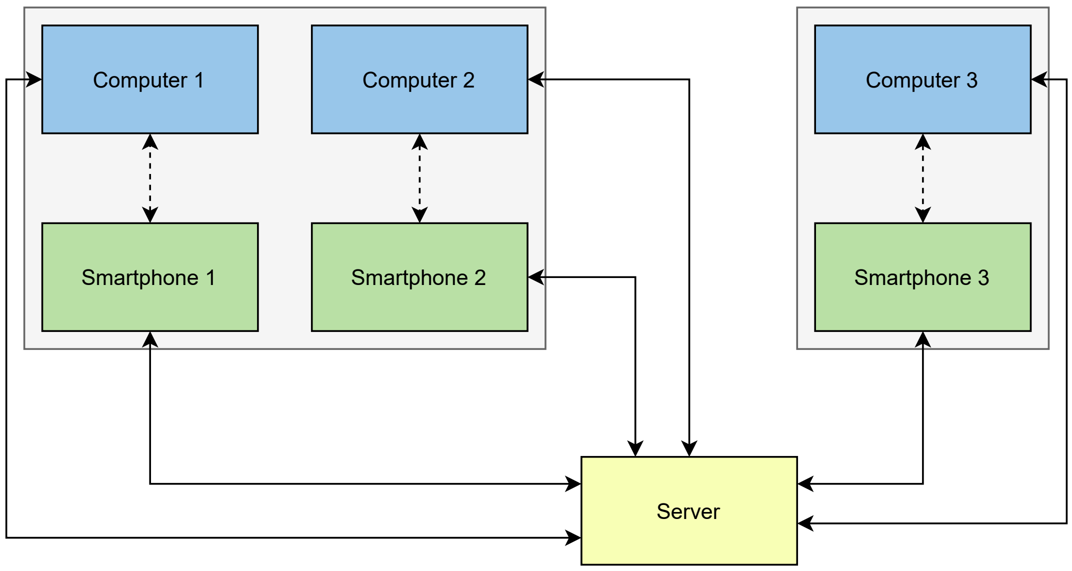


Figure 4.1: Deployment diagram

realized by saving each other's client ids and the relationship between the PCs is only saved in the player maps of the PCs (see chapter 6).

The system was tested mainly using an iPhone 6S and a MacBook Pro 2017, which also had the server running at the same time. As a browser, mainly Safari was tested. It does also work on other systems, but compatibility can not be guaranteed with every operating system and browser. Still, it is not confined to iOS or macOS devices but also runs on Android and Windows, respectively.

4.4 Comparison with Other Systems

Over the last few years, a lot of systems were created to do exactly what we want: capture movements and forward it to a computer that can then process the data and use it, for example, to play a game.

4.4.1 Kinect

Kinect is an add-on for the consoles of Microsoft that uses optical tracking to capture the user's movements. The first Kinect used a structured light approach whereas the second one uses a time-of-flight camera and infrared light. There have been sold almost 30 million of them, especially the first generation, but it was discontinued in 2017

[24]. It was not liked by the gamers as much as it inspired engineers to make new applications and try it in different environments [24]. For example, it can be used in a classroom to improve motivation and create an immersive learning experience [25]. Likewise, it is suitable as a 3D measuring device as shown in [26].

4.4.2 Nintendo Wii

The Wii by Nintendo was launched in 2006, which is a little earlier than the Kinect. It does not use an optical system but rather a hand-held controller with an integrated accelerometer to detect movement. Additionally, it works like a normal remote and can detect its pointing direction by sending a signal to the console itself [27].

The Wii, just like the Kinect, has been used for education and sports-based applications like in [28]. It has an advantage because there is also a balance board available that is used as an alternative controller and can aid in evaluating balance of people [29].

Even though devices like Kinect and Nintendo Wii have been accepted quite well by the users, the majority of households are still without such a system. We want to make it as accessible as possible and therefore use what almost everyone already has at home: a computer and a smartphone.

5 Software Prerequisites

5.1 Ubi-interact

For the interaction between the different devices in the system, we use a framework that was created specifically for the purpose of running on devices with different soft- and hardware. It does this by specifying the data formats, not the devices and their operating systems so the hardware can be easily changed without altering code [30]. This is very useful for a game that is supposed to run on a lot of platforms in order to make it available for as many users as possible. The basic functionality is illustrated in figure 5.1.

Ubi-interact works on a basic server-client architecture and sends data using the publish-subscribe pattern. The data is published under a unique topic which only the device that created the topic can send data on. Another device can then subscribe to this topic or multiple topics using a regex and receive the data that was sent. "Device" in this case is not necessarily a single device like a sensor or a laptop but just a collection of components [30].

In the basic case, a client sends the data to the server and another client receives it. But there is also the possibility of running a processing module on the server that transforms the data before passing it on. A processing module is a black box where you only define the input and output. This is helpful for using libraries that are not able to run on the client systems, handling a number of devices together without having to specify their number and it also helps make the code more re-usable. To bundle the processing modules, there are also sessions [31].

However, during the making of this thesis, these advanced functionalities were not usable because another important component, a *topic-muxer* was out of service. It would be responsible for bundling topics of the same kind so all the data that belong to one category can be received easily [30].

5.2 Vue.js

Vue.js is "a progressive framework for building user interfaces" [32]. It focuses on being incrementally adaptable and easy to learn. The main advantage for us is the reactive

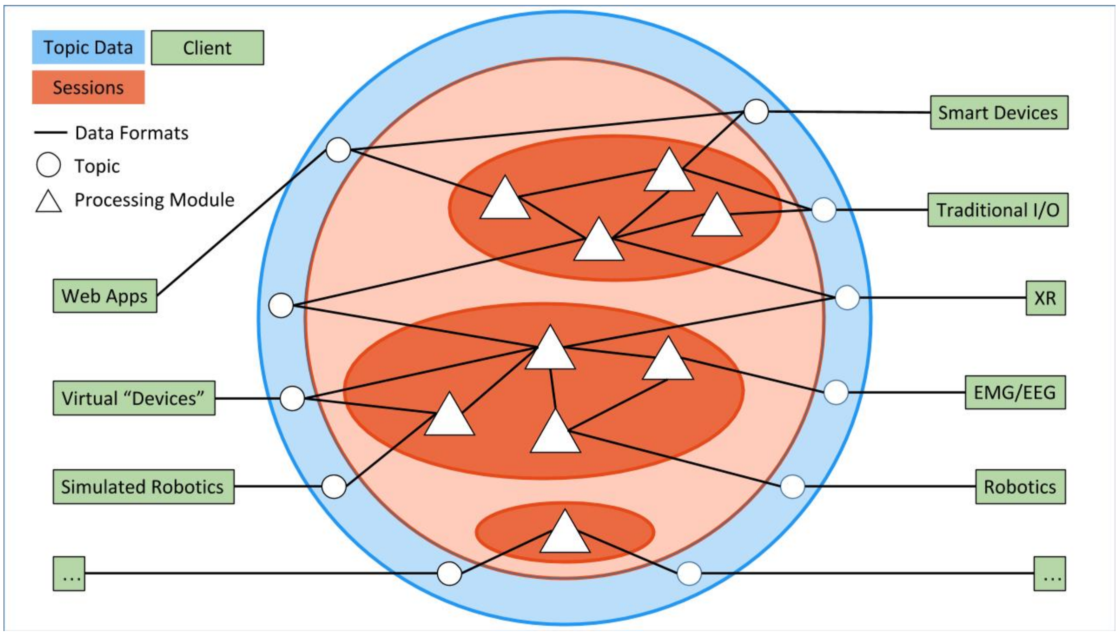


Figure 5.1: Overview over Ubi-interact, taken from [31]

components, but the conditionals and loops are also helpful [33]. Additionally, the user input is handled using Vue's *v-on* directive. The languages used are JavaScript, CSS and HTML.

5.3 Implementation

We chose to keep the application in the web browser and not make an app to simplify testing and deployment across larger distances as well as to make the game compatible with different operating systems. This system is also very easy to use for players as they only have to type in an URL and then follow the instructions on the display.

For the frontend, we use Vue.js and for the sending of the data, we use Ubi-interact. The frontend is not integrated into the Ubi-interact framework to make it easier to use.

To be able to access IMU data on iOS, it is necessary to use https or TLS-Encryption so we configured the Ubi-interact server accordingly. As already mentioned, the system was mainly tested using Safari as a browser but should also run on other platforms.

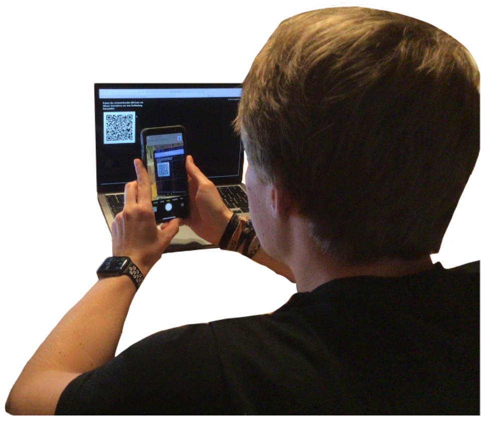


Figure 5.2: Image of a user scanning a QR-Code to establish the controller-desktop connection

5.4 Controller-Desktop

For our game to work, there needs to be a connection between the desktop and the phone of the player. To establish this connection as comfortably for the user as possible, we use a QR-Code. The user navigates to the entry page with their computer and scans the displayed QR-Code with their phone as seen in figure 5.2. This code leads to the mobile page and also contains the computer's client id given by Ubi-interact. The phone then publishes the desktop id on the topic "/clientId/handshake" so the desktop knows which client is its partner.

The players are connected with each other via the lobby system which is further explained in the next chapter.

6 Lobby System

6.1 Overview

To be able to have multiple groups playing at once, the players that are online need to be divided into groups. However, since Ubi-interact did not have its own lobby system yet, we had to make our own. This was done in a de-centralized way, because, as mentioned earlier, the topic-muxer was out of service. This is why the current states of all lobbies are saved on every client.

The lobbies are stored in two maps, one maps the lobby ids to their names (called lobby map) and the other one maps the lobby ids to another map of player ids mapped to player names (called player map). The player map is illustrated in figure 6.1. This way, all the data is easily accessible and displayable. The two aspects that need to be displayed are the list of lobbies the player could chose to join and the names of the other players that are currently in the lobby.

The user interface is in German and can be seen in figure 6.2.

The lobby id is always the client id of the host's computer, given by Ubi-interact. All of the lobby actions only have consequences for the computer of the users, not the smartphones. When the game is started, the current game state, including the lobby id is saved in an object to transfer the data to the next page and the mobile client is notified. There have to be three to eight players in the lobby to be able to start the game.

6.2 Actions

Basically, every time a client wants to do an action, this is published on a topic that contains the client's id. All other clients are subscribed to the topic via a regex. The client publishes on "/clientId/<what to do>" to make sure that the topic is always unique. In the object that is passed, more information is added, like which lobby to join or the name of the player. For example, when a player wants to join a lobby, the client publishes the desired lobbyId and their name on the topic "clientId/join_lobby". Because all other clients are subscribed to the regex "/*/join_lobby", they know the player wants to join.

The actions are: *greet*, *make lobby*, *join lobby*, *leave lobby* and *start game*. When a client

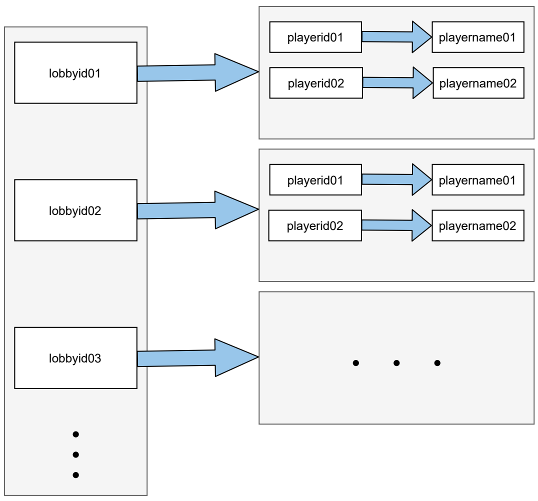
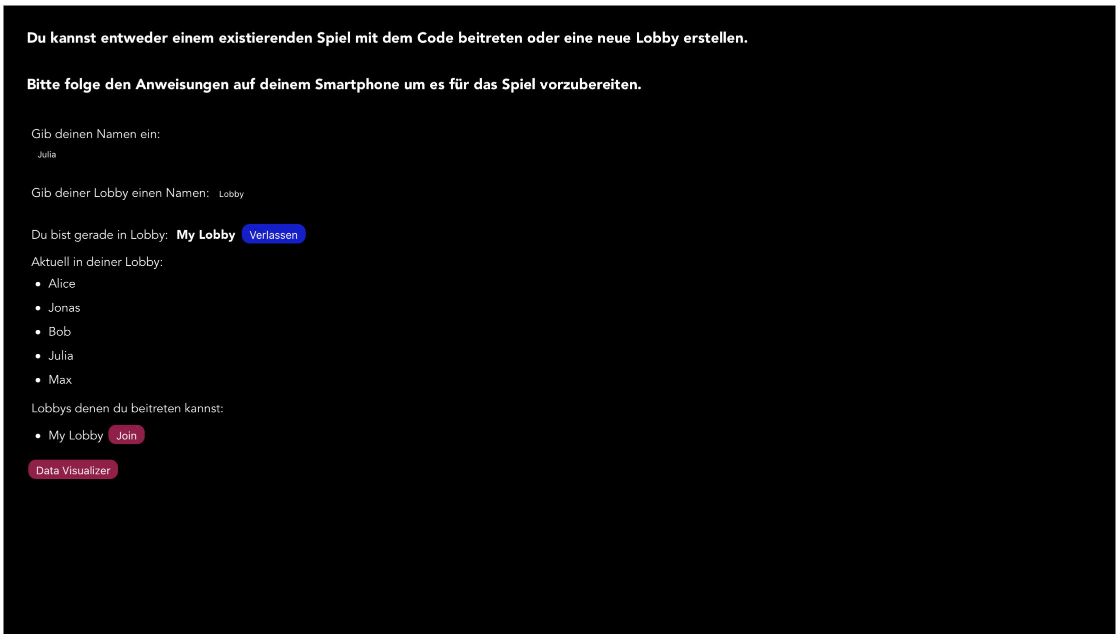


Figure 6.1: Illustration of the data structure used to save all lobbies



Du kannst entweder einem existierenden Spiel mit dem Code beitreten oder eine neue Lobby erstellen.

Bitte folge den Anweisungen auf deinem Smartphone um es für das Spiel vorzubereiten.

Gib deinen Namen ein:
Julia

Gib deiner Lobby einen Namen: Lobby

Du bist gerade in Lobby: **My Lobby** Verlassen

Aktuell in deiner Lobby:

- Alice
- Jonas
- Bob
- Julia
- Max

Lobbys denen du beitreten kannst:

- My Lobby Join

Data Visualizer

Figure 6.2: Screenshot of a lobby with five players

receives a *greet* signal and it is the host of a lobby, it sends its lobby id and lobby name. This makes it possible for players to join lobbies that were made before they connected.

When receiving a *make lobby* command, a new lobby is added to the lobby map and the lobby id is also added to the player map, after making a new map with the host's id and name in it to provide the content for the lobby in the player map.

When other players want to join an existing lobby, the *join lobby* command is used. Upon getting this message, a client simply adds the player's id and name to the correct lobby in the player map.

Finally, it can also happen that a player wants to leave a lobby and join another one or quit the game. For this, we use the *leave lobby* command. If the player leaving is the host, the lobby will be deleted. This is done by removing this lobby id from both maps. All other players that were in this lobby will be thrown out. If it is another player leaving, this player id and -name will just be deleted from the player map.

To save important information while changing the page, we created a singleton object, the GameState object, to hold the data for later. It is accessible from every page and is unique for each client.

When the host starts the game, the *start game* signal is sent along with the current GameState object of the host. To continue on to the ingame-screen, some data needs to be saved first. Most importantly, the map of players currently in the lobby needs to be saved. It is converted to an array of objects that also includes the current points and direction for the visual representation of the movement later. Additionally, we need to save the rotation the smartphone was calibrated to earlier and the lobby id. The host device randomly determines a faker and saves it in its GameState object before sending it out so all the other clients can save it in theirs. Once all the data is saved successfully, the player is re-directed to the ingame-page and the video automatically starts playing the dance.

After all changes to the maps it is important to make a new object instance of it to force Vue to re-render this part of the page and update the lists correctly.

6.3 Errors

The system is built in a way to avoid errors produced by the user, this means that buttons only appear when the user is permitted to press them. For example, the "leave lobby" button only exists if the player is already in a lobby and the "make lobby" button disappears once the player joins another one. For the remaining scenarios, there are error messages in place that let the user know what they did wrong and why the desired action is not possible at the moment.

The problem is that, because the state of all lobbies needs to be saved on all clients, the system is not very scalable. This could however easily be fixed by copying the code into a processing module that runs on the server and make some minor adjustments. This would also fix the remaining issue that, when a new client connects, only the list of lobbies they could join is passed to them, but some other information is missing. The centralized version could handle this very easily and can be re-used by future applications.

7 Gesture Recognition

There are a lot of approaches to recognize gestures algorithmically which can be classified into three groups: template-based, statistical-based and machine learning-based [34].

7.1 Gesture Classification Approaches

7.1.1 Template-based approaches

Template-based means that there is a catalogue of gestures and data on how these should look like. There are then different ways to compare these to the input the user delivers and match the most likely gesture. The drawback of this method is that you need to find one generic dataset to save in the catalogue as example. This can be hard for more complicated gestures because they might have quite different data depending on the person [34].

Euclidean Distance Metric

Calculating the euclidean distance metric is one of the simplest ways to detect gestures. You just calculate the difference between the vector saved in the catalogue file and the input. Afterwards, the gesture with the smallest deviation is selected as the correct answer. It is used for instance in [34].

7.1.2 Machine Learning

Machine learning has gained a lot of popularity during the last years. It is especially helpful to use in cases where we have noisy data that need to be classified. It is popular in gesture recognition because you do not need to worry about issues like, for example, the same length of gestures, as much.

To use a machine learning model, you need to train it first. For this, you need data from multiple people to avoid over-fitting which is a problem where the model is so well fitted to one person, it can not accurately detect gestures of other people because it is too specialized. Each person needs to perform the gesture many times to train the model well, which is a lot of effort.

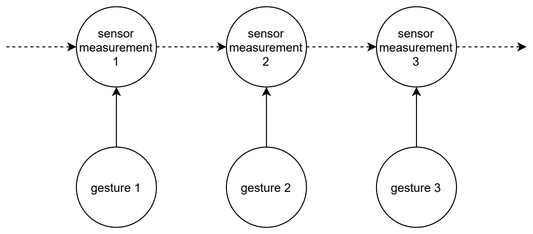


Figure 7.1: Illustration of a Hidden Markov Model

When machine learning is used, Tensorflow is oftentimes the first choice because it can be used with JavaScript and it is easy to understand. This has for example also been chosen in [35] to classify gestures measured by EMG Sensors.

7.1.3 Hidden Markov Models

Hidden Markov Models are normally used in statistical-based approaches [34]. They consist of hidden states and a process depending on these. The hidden states, in the case of gesture recognition, correspond to the gesture performed and the process represents the data we get from the sensors. So we can not observe which gesture was performed, but it influences the sensors' measurements, which we can detect [36]. This relationship can be seen in figure 7.1.

In short, we use HMMs (one for each gesture) to predict the next time step based on the user input until now and the prior training, and classify the gesture accordingly. You need to train the models beforehand just like you do with machine learning which is a drawback because of the different people required and the time effort. Undoubtedly, they do give good results in most cases.

After trying some approaches, [4] came to the conclusion that Hidden Markov Models (HMMs) are the best way to classify tennis strokes. This shows that this classification technique is a good fit for smartphone IMU applications.

They were also used in [37] and provided a very high classification rate for session-independent data. However, for person-independent, it was only 74.3% so there needs to be a calibration for every user to make it work reliably.

7.1.4 Finite State Machines

Less commonly used, yet very interesting, is another approach that uses finite state machines to classify the data [38]. The machine is created using training data and k-means. The state with the largest variance is split when the error improvement gets too low. This is repeated until all state's variances are under a certain threshold. Later on, they manually specify the temporal ordering to get the finished state machine [38].

This is different from HMMs because the training data does not need to be well-aligned and the structure of the states is not defined. Additionally, the computational complexity is smaller and the classification therefore faster [38].

7.2 Our Approach

7.2.1 Simplifications

We have the advantage that since we know the dance, we know which move the player is supposed to be doing at the moment. We also know the duration of the movement so we do not have the problem of finding the beginning and the end of the gesture, we just take the time frame of when the gesture is supposed to be performed.

We also do not need a very accurate gesture recognition where we are able to distinguish between gestures and there is only a right or a wrong answer. What we need is a scoring system on how well the player performed what they saw on the screen as well as a way to display what the others did.

7.2.2 Globalization of Acceleration

All the data we get from the phone is given in a local coordinate frame that can be seen in figure 7.2. Because we want to know in which direction the phone is moving relative to the computer screen, we need to transform the acceleration into this different coordinate system. Our global coordinate system is oriented relative to the screen, or better said relative to the starting, calibration position of the phone.

To do this, we just use the orientation we are given and rotate in the opposite directions. Because of how the angles are given in this API, we need to rotate around z-x-y in this order. So we take the rotation matrices for the angles and multiply them in this order to get:

$$\begin{gathered}\begin{pmatrix} cos(z) & -sin(z) & 0 \\ sin(z) & cos(z) & 0 \\ 0 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(x) & -sin(x) \\ 0 & sin(x) & cos(x) \end{pmatrix} * \begin{pmatrix} cos(y) & 0 & sin(y) \\ 0 & 1 & 0 \\ -sin(y) & 0 & cos(y) \end{pmatrix} = \\ \begin{pmatrix} cos(z)*cos(y) - sin(y)*sin(x)*sin(z) & -cos(x)*sin(z) & cos(z)*sin(y) + cos(y)*sin(x)*sin(z) \\ sin(z)*cos(y) + sin(x)*cos(z)*sin(y) & cos(x)*cos(z) & sin(z)*sin(y) - sin(x)*cos(z)*cos(y) \\ -cos(x)*sin(y) & sin(x) & cos(x)*cos(y) \end{pmatrix}\end{gathered} \tag{7.1}$$

For x,y and z, we need to plug in the negated values of the (local) rotation around the respecting axes.

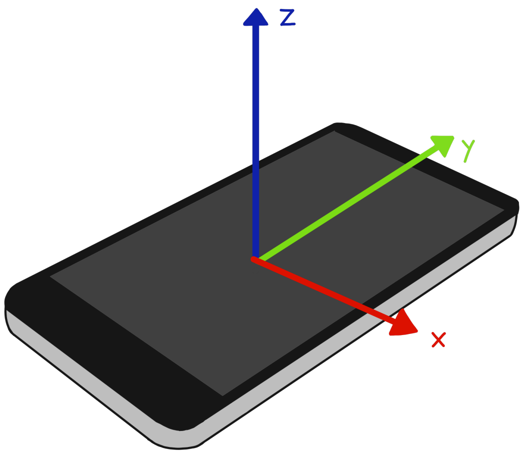


Figure 7.2: Illustration of the coordinate system of the smartphone

To get the correct relative orientation, we need to calibrate the phone at the beginning. For this, we ask the player to make the arrow on their smartphone point to their PC. Then, they should press a button and the phone publishes its current position to their desktop. This orientation also needs to be saved in the GameState object to be accessible from every page.

7.2.3 Two Approaches

With the assumptions and the globalised acceleration, we developed two approaches: For the first one, we only take the direction of the movement in the global coordinate system and classify it into one of six categories: up, down, right, left, front or back. Although this is not very accurate, it is robust and simple. We use this to show the players arrows for each participant to visualize the movement.

Direction Arrows

To smooth the data, we first filter out small acceleration values and put the remaining ones into a round buffer. We then iterate over the buffer and sum up the contents to get an estimate of the velocity. To be accurate, we would need to multiply this with the time difference but this is not necessary in our case because the time difference to the last measurement is always the same and we would therefore just scale the data.

Once we have obtained the velocity, we sum up all of the components of the vector and discard it if it is too small. Afterwards, we calculate the difference between the axes, so x-y, x-z and y-z. These values are used to get the axis on which the movement occurred, for example the y-axis. To do this, we compare two of the difference values, in our case $diffXY$ and $diffYZ$ to the threshold value $minDifference$ in the following way:

$$diffXY < -minDifference \text{ \&\& } diffXZ > minDifference$$

Like this, we only need three difference values to compare all of our axes against each other. Afterwards, the direction on the axis is determined by the sign of the velocity estimate, for example "right" if the sign is positive.

Because the acceleration is very unstable and oscillates around zero, the velocity does as well. The averaging helps with this problem because the negative and the positive values cancel each other out. With the addition of the filtering of small values, we get filtered data.

In the end, we publish the calculated direction so the other players' devices in our lobby can display it.

Points by Euclidean Distance Metric

For the second approach, we calculate the euclidean distance between the vector of the recorded dance and the data the player is producing and calculate a score based on this. For this, we use the acceleration, orientation and velocity. This data is first recorded and stored in a CSV file, which is then read at the start of the game.

We compare each category (orientation, acceleration, velocity) against each other in every time step, so every 10 ms. For this, we iterate over the round buffer and average each category. For the acceleration and velocity, we just add the components of the vectors together and divide them at the end. After the loop, we take the distance vector between the expected and the actual data, calculate its length and give points based on how much it is off. Of course, a small deviation is normal and therefore not punished.

Because the orientation is given in Euler Angles, we can not just sum them up normally. We need to take into account that 359 and 0 is only one degree apart. The orientation of the DeviceOrientationEvent [39] is given in the following intervals: $x \in [-180, 180), y \in [-90, 90), z \in [0, 360])$. To get the smallest difference between two measurements, we use the following formula: $min((m) - |a - b|, |a - b|)$ where m is the upper bound of the respecting interval and a and b are the two angles.

Because we need to do this conversion, we can not use the same approach as for the other two measurements. Instead, we calculate the differences and then add them into one variable. This is then divided after the loop to make an average and we can give points based on the value.

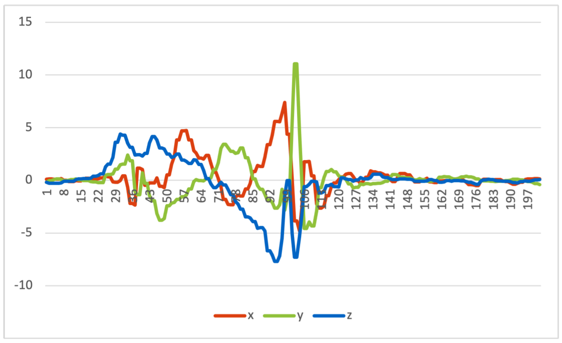


Figure 7.3: Graph of the acceleration for a badly performed up-gesture

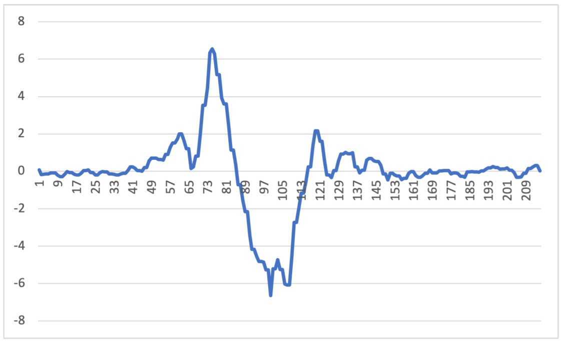


Figure 7.4: Graph of the acceleration during a single movement along one axis

At the end of the function, the first value in the buffer is deleted and in the next pass-through, the next number of values is compared.

This comparison method would work with more complicated gestures as well.

Difficulties

It is hard to move a smartphone along one axis and one axis only. Every deviation from the original path is an acceleration and makes it harder to determine the actual axis the smartphone was moved along. An example of a shaky and lazily performed "up" gesture can be seen in figure 7.3. The result of the classification should be "up", so positive z, but because of all the shaking this is almost impossible to see from the data. Additionally, the movement was performed lazily and the acceleration in the correct direction is not very big as a result.

Another issue is that when a player performs a motion along one axis in positive direction, the acceleration is positive first, then negative when they slow down the movement because the arm reached its maximum extension. The graph of this movement can be seen in figure 7.4. This leads to the problem that we can not easily distinguish if an acceleration along the negative axis is from slowing down or because the player started a new movement in the other direction.

Even though we filter the data already, it is still quite noisy. If we sum the velocity up over time like this: *velocity = velocity + acceleration * timedifference* it never gets back to zero, even after only one movement. This shows that the positive and negative acceleration do not cancel each other out sufficiently and there have to be errors that accumulate over time.

8 Results

8.1 Player Satisfaction

8.1.1 First Test

At the very beginning of this work, a small trial run was made to find out the potential of the game and how to improve the idea. There were five people in this test. They were connected via a discord-session and could see each others' cameras. Everyone was sent a video, one person got a sightly different version but did not know so. Now all test subjects started their videos at the same time and did the dance as best as they could. At the end, everyone had to guess who the faker was. This was done in a conversation so everyone gave a reasoning for their answer. Even though the right person was found, one could see from the reasons they gave that it was more a coincidence.

It was found that it is quite hard to do a dance for the first time and do it accurately enough so that other people can tell subtle differences. It is part of the game to figure out if a player is doing something differently on purpose or if it is just a mistake, however it was a little extreme in this case. We concluded that we needed to make the dance easier and more repetitive to give the players a chance to learn the moves and do them while looking at the others rather than at the model video.

The biggest point of criticism was that the faker did not know it was them. This was consequently changed for the final version. The players also made a few suggestions that would make the game more interesting and could be introduced as different modi later.

All in all, the players enjoyed themselves and said they would play it again sometime.

8.2 Gesture Classification Accuracy

The calculation of the points is hard to judge in numbers, it is more of a subjective rating if it "feels right" to the player. Additionally, because the checking is done continuously for the whole duration of the dance and not per gesture, it is difficult to determine what the system exactly gave points for and what not. Also, the game does not give any other direct feedback like words of affirmation on the screen because it would be distracting and too hard to watch the words while still paying attention to the rest

Table 8.1: Classification rates of the two versions for direction detection
CDAR: correct direction was amongst the results, SDP: only slowing down problem, Version 1 disregards fast movements

	Correct	CDAR	SDP	Wrong
Version 1	46.15%	23.08%	0%	30.77%
Version 2	21.62%	21.62%	32.43%	24.32%

of the players. Consequently, it is questionable if the players even noticed the points frequently enough to judge their fairness.

Some sanity checks, like not moving the smartphone at all or doing the dance of the faker as an inspector, were concluded and showed positive results. To determine if the players feel it is fair, further tests would have to be done.

In contrast, the direction classification was not as easy to do as anticipated. The slow movements can be classified correctly in 46.15% of the cases and the correct direction is displayed alongside another, wrong direction, in an additional 23.08% of the cases in the final dance. The actual classification accuracy is lower than that because the gestures that worked better were used in the dance and faster movements can not be detected at all.

We will now look into the details of how we tried to overcome the difficulties.

Because of the issue with slowing down that was mentioned earlier, the classification result is often times "up - down" for a single movement upwards.

We tried to fix this issue by averaging over a larger circular buffer and by checking the buffer less frequently. However, this makes it less responsive and harder to classify faster movements. One would think that it is possible to find a middle way but checking every 13 measurements, which is every 0.13 seconds, makes it jitter and we have the slowing down problem and if we only check every 14 times, so every 0.14 seconds, it makes it laggy and unable to recognize fast movements.

The approach with the jitter problem is good at detecting fast movements because if the player starts a downwards motion right away every time they finish their upwards motion, the slowing down is the same as the acceleration in the new direction and therefore the problem is not visible. It gets 21.62% completely correct, 16.22% of the results at least contain the correct direction and 32.43% are only wrong because they include the slowing down as a direction. All the classification rates of the two versions can be seen in table 8.1.

We tried to get rid of the slowing down problem by saving the last recognized direction and then only putting "down" as a valid classification if "up" was not the last direction and so on. This however only resulted in "up-down" changing to "up-none-down" so we would need to account for this for a longer time but then, fast direction changes on the same axis can not be detected any more.

In the end, we decided to go with the laggy version and just not put a great emphasis on the fast movements in the dance. The quicker steps are only there to make it more challenging, but the faker has to be found using the slower steps. The points are not influenced by the direction recognition in any way so it is not possible for one player to just not do the fast steps and be lazy.

For the buffersize, we decided on 130 for the direction buffer and 20 for the points. The big discrepancy comes from the problem just described that forces us to use a bigger buffer for the direction recognition. We could not make it even bigger because 130 measurements already hold the data of 13 seconds and the gestures are performed in less time than that. But if we make it smaller, we do not have enough smoothing.

We found that, generally, the measurements are quite inconsistent. One could do the same motion twice and get quite different results. For example, in figure 8.1 the same person performed an up-gesture twice, right after one another, and the data looks quite different even though the detection is still possible. If there is a change of person, this is obviously more extreme. Once again this illustrates how important smoothing and filtering is.

8.3 Kalman Filter

The main improvement that should be done is using a Kalman Filter. Hopefully, this would increase the accuracy by providing better data to work with.

To make the Kalman Filter, we need a motion model which would just be the basic motion equations in our case.

It works in two steps: the first one is the predict-step and the second one is the update-step, together they have five equations. In the predict step, we use the control matrix and a state transition matrix to predict the current state and the error covariance.

Afterwards, in the update-step, we calculate the Kalman Gain and the error covariance is updated. Then, the state estimate is updated using the measurement vector [40].

8.4 Future Work

The arrows could be replaced by glowing streaks to visualize the movement better and allow for more different gestures. Because it is not desirable to calculate the position of the smartphone, we propose an approach where the gesture is classified and based on this classification, an animation is played. So the animation is not exactly what the

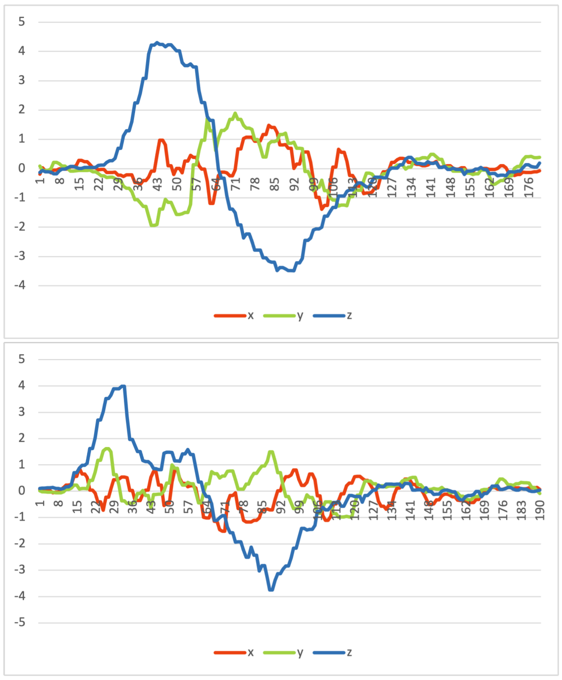


Figure 8.1: Data of the same person doing the same movement

player does but what the classifier thinks the player is doing. An example of how that could look like can be seen in 8.2.

To improve the classification itself, we could use a HMM or Tensorflow if we can provide the data needed to train them. This should improve classification enough to base both the points and the visualisation on it.

Furthermore, we could also improve the gameplay to make it more variable and interesting. The players in the first test for instance suggested to not give the faker any video to copy from but just let them try to copy the inspectors. Then, a short timespan of darkness could be introduced to make the faker struggle and have to improvise to give the inspectors a chance.

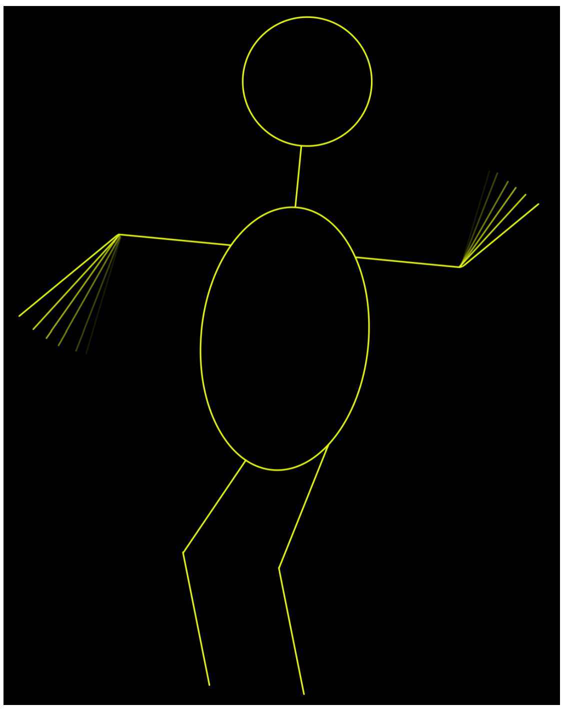


Figure 8.2: Example of a different visualization

9 Summary

To summarize, we created a game that has sportive and social deduction elements. The game is played with a smartphone as a controller and a PC for all visual elements.

We wrote a lobby system for Ubi-interact to connect the players with each other in groups and make sure the communication between the devices runs smoothly. Additionally, we created a way to record the IMU data, save it into a CSV file and download it.

For the recognition, we implemented two different approaches: one for the visualization of the players' movements and one for the calculation of points given. The points work well, whereas the directional approach has some challenges that are still present.

To filter the data for the directional recognition, we discarded small values from the acceleration while summing up over a buffer. Then, we added all components of the vector together. If this number was too small, we discarded it, otherwise we calculated the differences between the axes to determine the axis the movement occurred along. We found the biggest of these difference values, accounting for sufficient space between them. Finally, we used the sign of the acceleration sum along the axis we just discovered to determine the direction. In the end, we got a 69.23% recognition rate in slow gestures with this approach.

For determining the points, we did not do as much smoothing. We only averaged over the, much smaller, buffer without filtering the small values first and used these to determine the difference between the user input and the recorded gesture. The reason for this is that it is not needed because the recorded data is not filtered either and it does not need to be as accurate, we can just put the threshold of classifying it as "correct" a little higher and save some computation time.

In the end, we get a playable demo that still needs some work before it is ready to be played as a party game for people that like to exercise and play together but can not do so because of various reasons.

List of Figures

List of Tables

Bibliography

[1] A. Kirk, A.-M. Knowles, and A. Hughes, "Is sitting bad for your mental health?," May 2014.

[2] D. Bavelier and C. Green, *The brain-boosting power of video games on jstor*, Accessed On: 03.09.2021. [Online]. `https://www.jstor.org/stable/26047025`, Jul. 2016.

[3] P. T. Alpert, "The health benefits of dance," *Home Health Care Management and Practice*, vol. 23, pp. 155–157, 2 Apr. 2011.

[4] M. Mitschele, "Determining valid movements with pseudotracking for augmented reality tennis," 2019.

[5] A. Y. Benbasat and J. A. Paradiso, "An inertial measurement framework for gesture recognition and applications."

[6] "Feasibility of wrist-worn, real-time hand, and surface gesture recognition via semg and imu sensing," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 3376–3385, 8 Aug. 2018.

[7] N. Siddiqui and R. H. Chan, "Multimodal hand gesture recognition using single imu and acoustic measurements at wrist," *PLoS ONE*, vol. 15, 1 Jan. 2020.

[8] A. S. Kundu, O. Mazumder, P. K. Lenka, and S. Bhaumik, "Hand gesture recognition based omnidirectional wheelchair control using imu and emg sensors," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 91, pp. 529–541, 3-4 Sep. 2018.

[9] N. ′. Brandizzi, L. Iocchi, and D. Grossi, "Rlupus: Cooperation through emergent communication in the werewolf social deduction game."

[10] N. Vinod, *How many people play among us?* Accessed On: 12.09.2021. [Online]. `https://www.sportskeeda.com/esports/how-many-people-play-among-us`, 2020.

[11] S. Tilton, "Winning through deception: A pedagogical case study on using social deception games to teach small group communication theory," *SAGE Open*, vol. 9, 1 Jan. 2019.

[12] M. Zhang, "Speech comprehension and role prediction in the game of werewolf," 2018.

[13] R. Clements, *Dancemasters kinect review*, Accessed On: 09.09.2021. [Online]. `https://www.ign.com/articles/2010/11/09/dancemasters-kinect-review`, 2012.

[14] U. Entertainment, *Just dance now*, Accessed On: 08.09.2021. [Online]. `https://justdancenow.com`.

[15] R. Topala, *Geometry dash*, 2013.

[16] C. M. Inc., *Piano tiles 2*, Accessed On: 11.09.2021. [Online]. `https://cheetahgames.cmcm.com/PianoTiles2`, 2018.

[17] U. Entertainment, *Just dance 2021*, Accessed On: 08.09.2021. [Online]. `https://www.ubisoft.com/de-de/game/just-dance/2021`.

[18] H. Ahmed and M. Tahir, "Improving the accuracy of human body orientation estimation with wearable imu sensors," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, pp. 535–542, 3 Mar. 2017.

[19] D. S. Tran, N. H. Ho, H. J. Yang, E. T. Baek, S. H. Kim, and G. Lee, "Real-time hand gesture spotting and recognition using rgb-d camera and 3d convolutional neural network," *Applied Sciences (Switzerland)*, vol. 10, 2 Jan. 2020.

[20] A. Kadambi, A. Bhandari, and R. Raskar, *3d depth cameras in vision: Benefits and limitations of the hardware with an emphasis on the first-and second-generation kinect models*, 2014.

[21] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, and V. Kasi, "Reviews on various inertial measurement unit (imu) sensor applications," *International Journal of Signal Processing Systems*, pp. 256–262, 2013.

[22] T. Gadeke, J. Schmid, M. Zahnlecker, W. Stork, and K. D. Muller-Glaser, "Smartphone pedestrian navigation by foot-imu sensor fusion," 2012.

[23] Y. Qifan, T. Hao, Z. Xuebing, L. Yin, and Z. Sanfeng, "Dolphin: Ultrasonic-based gesture recognition on smartphone platform," IEEE, Dec. 2014, pp. 1461–1468.

[24] M. Weinberger, *The rise and fall of kinect: Why microsoft gave up on its most promising product*, Accessed On: 07.09.2021. [Online]. `https://finance.yahoo.com/news/downfall-kinect-why-microsoft-gave-183900710.html`, 2018.

[25] J. H. Hui-Mei, "The potential of kinect in education," *International Journal of Information and Education Technology*, vol. 1, 5 2011.

[26] J. Smisek, M. Jancosek, and T. Pajdla, *3d with kinect*, 2013.

[27] M. Morgan, M. Butler, and M. Power, "Evaluating ict in education: A comparison of the affordances of the ipod, ds and wii."

[28] Y. Y. Chao, Y. K. Scherer, and C. A. Montgomery, *Effects of using nintendo wii™ exergames in older adults: A review of the literature*, Apr. 2015.

[29] R. A. Clark, B. F. Mentiplay, Y.-H. Pua, and K. J. Bower, "Reliability and validity of the wii balance board for assessment of standing balance: A systematic review," *Gait and Posture*, vol. 61, pp. 40–54, 2018.

[30] S. Weber, D. Dyrda, M. Ludwig, and G. Klinker, "Ubi-interact," Association for Computing Machinery, Dec. 2020, pp. 291–300.

[31] S. Weber, *Ubi-interact wiki*, Accessed On: 07.09.2021. [Online]. `https://github.com/SandroWeber/ubi-interact/wiki/Overview`, 2020.

[32] E. You, *Vue.js*, Accessed On: 07.09.2021. [Online]. `https://v3.vuejs.org/guide/introduction.html`, 2021.

[33] P. So, *Vue.js*, 2018.

[34] B. J. Kim, J.-i. Jung, and S. H. Park, "Recognition of multiple concatenated arm gestures using six-axis inertial sensor signals," *International Journal of Control and Automation*, vol. 11, pp. 25–36, 5 May 2018.

[35] A. Pomelova, "Classification of semg using tensorflowjs in ubi-interact-playing rock-paper-scissors," 2019.

[36] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, vol. 3, pp. 4–16, 1 1986.

[37] M. Georgi, C. Amma, and T. Schultz, *Biomedical Engineering Systems and Technologies*, A. Fred, H. Gamboa, and D. Elias, Eds. Springer International Publishing, 2015, vol. 574.

[38] P. Hong, M. Turk, and T. S. Huang, "Gesture modeling and recognition using finite state machines," 2000.

[39] Devices and S. W. Group, *Deviceorientation event specification*, Accessed On: 05.09.2021. [Online]. `https://w3c.github.io/deviceorientation`, 2021.

[40] P. Balzer, *Das kalman filter einfach erklärt [teil 2]*, Accessed On: 12.09.2021. [Online]. `https://www.cbcity.de/das-kalman-filter-einfach-erklaert-teil-2`, 2013.