FACULTY OF ENGINEERING TECHNOLOGY

**CAMPUS GROUP T LEUVEN**

# Design and implementation of an application using motion input for controlling computer games as part of physical therapy

**Christiaan VANBERGEN**

**Dimitri VARGEMIDIS**

Supervisor: prof. dr. ir. L. Geurts

Co-supervisor: ir. K. Vanderloock

Master Thesis submitted to obtain the degree of
Master of Science in Engineering Technology:
Electronics-ICT, Internet computing

Academic Year 2016 - 2017

# Acknowledgements

Designing and implementing a user interface utilizing the relatively new Kinect camera is a challenging, albeit fascinating subject. We encountered many problems along the way, for which we enjoyed coming up with creative solutions. In the process, we were supported by many people we definitely would like to thank.

This thesis would not have been possible to write without our supervisor professor Luc Geurts and co-supervisor Karen Vanderloock from e-Media lab, who provided us with this subject, invested their time and shared their knowledge and experience with us, allowing us to come up with solutions that have lead to a fully functional result.

During the year, we've had several occasions to discuss the subject of machine learning and support vector machine (SVM) with our fellow students Victor Martens and Alexander Kerckhofs, who helped with making us aware of the pitfalls and shortcomings of the used techniques and joined us to discuss the requirements and come up with a creative solution concerning the use of SVM.

User testing is very important when designing an application with a specific audience in mind. As such, we are grateful for the useful feedback of physical therapist Dries Lamberts of Windekind, who took the time to meet us twice and evaluate both our prototype as well as the final result. But even more importantly, he shared with us his passion for taking care of children with disabilities, their need for physical exercise and the importance of making sure this is both meaningful and entertaining.

Last but not least, we would also like to thank our parents, family and friends, who not only have supported us during the course of past year while working on this master's thesis, but also for trying out our application and giving valuable feedback.

# Abstract

Using games as part of physical therapy improves patient motivation to perform the necessary exercises. The developed application allows physical therapists to choose exercises that fit the needs of the patients and record them using the Kinect 2.0 camera. When the patient executes an exercise, machine learning algorithms are used to recognize the exercise, which activates the keyboard key linked to that exercise. This makes it possible to interact with any available game relying on key inputs.

Research is conducted on the subject of human-computer interaction (HCI) involving the use of the Kinect camera to interact with the application in what is called a natural user interface (NUI). Several prototypes are designed that focus on different interaction principles and are divided into two categories: the mime pattern and the hints pattern. The chosen implemented prototype of the interface allows users to interact with the menus and items appearing on-screen without the use of traditional buttons. Interactive elements like pulling ropes are present in the interface, providing both feedforward and feedback to the user in an effort to simplify the interaction process.

As part of the user-centered design approach, the interface is evaluated by conducting a user test with a physical therapist. Feedback is obtained both during the prototyping phase, as well as after having implemented the prototype. A conclusion from this test is that it takes more time to understand how some elements can be interacted with. With a short explanation, it is possible to decrease the time required to achieve this. The application as a whole has a low learning curve as the focus on simplicity results in the user being familiar with all required actions after recording an exercise once.

alternative abstract

Patients are often demotivated by pain or boredom during physical therapy and previous attempts at using motion controlled games to improve this worked, but had limited exercises and monotonous game play that proved to make it economically inviable. In this thesis a more versatile program is developed by using the kinect 2.0 3D-camera to allow a physical therapist to train a support vector machine (SVM) algorithm to recognize any exercise and link it to an action in any game that uses keyboard input. Though this thesis is more focused on the human-computer interaction (HCI) aspect of designing a user-friendly motion controlled natural user interface (NUI) for this program.

For the development of the NUI several special paper-prototypes are designed that focus on different interaction principles. After user feedback a principle was chosen and further developed into a coded prototype that uses straight-forward interaction with good feedforward and feedback in an effort to streamline the interaction process. After a final user test, small changes to the prototype were applied. For the recognition of exercises a few choices had to be made about the processing of the input data.

The final conclusion is that most users recognize the required interactions fairly quickly and are able to efficiently control it rapidly.Though there were some instances were the user expected a different reaction after an action. It was obvious that the application will need some additional information to teach the user how to record good exercices. During recognition several positions or movements can be distinguished

with good accuracy.

Key words: exergames, gesture recognition, Kinect 2.0, physical therapy, support vector machine, user interface

# Extended abstract

Kinesitherapie is vaak essentieel bij revalidatieprocessen of om de spieren te trainen voor mensen met lichamelijke beperkingen. Computerspellen kunnen deze sessies en bijhorende oefeningen aangenamer maken om uit te voeren en betekenen een belangrijke vorm van motivatie, in het bijzonder voor kinderen. Het probleem met deze spellen is dat ze duur zijn om te ontwikkelen en aan te kopen en dat de oefeningen die een bepaald spel aanbiedt op voorhand vastliggen. Hierdoor zijn deze spellen niet altijd zinvol en effectief om te gebruiken bij kinesitherapie.

De ontwikkelde applicatie biedt meer flexibiliteit en laat toe om reeds bestaande spellen te spelen door middel van bewegingen. De applicatie is gericht op kinesisten en laat hen toe om bewegingsoefeningen op te nemen die zinvol zijn voor de patiënt. Het is dan mogelijk om naar eigen keuze een knop van het toetsenbord toe te kennen aan elke opgenomen oefening. Wanneer de patiënt vervolgens deze oefening uitvoert, herkent de applicatie deze beweging en wordt de knop die eraan toegekend is virtueel ingedrukt. Zo is het mogelijk om met bewegingen eender welk spel te spelen dat gebruik maakt van toetsenbordbediening.

De kinesist kan oefeningen opnemen met behulp van de Kinect 2.0-camera. De applicatie gebruikt machine learning en support vector machine (SVM) om deze oefeningen aan te leren. Het geïmplementeerde algoritme splitst elke beweging op in een aantal kleinere deelbewegingen en classificeert deze elk onder een andere label. Tijdens het spelen herkent het algoritme een beweging als alle deelbewegingen ervan zijn waargenomen in de juiste volgorde. Deze benadering laat toe om de rekenkost van bewegingsherkenning te beperken en zorgt voor de patiënt voor een betere responsietijd na het uitvoeren van een beweging.

Om bewegingen op te nemen, maakt de kinesist gebruik van de grafische gebruikersinterface. Het is mogelijk om interactieve elementen van de initiële prototypes onder te verdelen volgens twee categorieën: mime-elementen en hints-elementen. Mime omvat elementen die herkenbaar kunnen zijn uit het dagelijks leven, zoals deuren, handvaten en trekkoorden. Omwille van hun herkenbaarheid heeft de gebruiker reeds een notie van hoe interactie mogelijk is met deze elementen, waardoor het leerproces minder groot is voor de gebruiker en hij minder moet onthouden wat betreft manieren van interageren. Hints-elementen verwachten handelingen van de gebruiker die niet teruggrijpen naar objecten uit het echte leven, zoals een scrollbars met bijhorende veegbewegingen om ermee te interageren. Na gebruikerstesten met een kinesist is een prototype weerhouden dat berust op een combinatie van zowel mime-elementen als hints-elementen.

Bij de uitwerking van dit prototype ligt de nadruk op een interface die de acties van de gebruiker koppelt aan de functies van de interactieve elementen. Met andere woorden reageren deze elementen op de manier die de gebruiker verwacht. Zo bewegen ze met dezelfde snelheid waarmee de gebruiker zijn hand beweegt over een element en reageren ze op interacties door bijvoorbeeld van kleur te veranderen. Daarnaast reageert de interface, perceptueel gezien, onmiddellijk op gebruikersinteractie met grafische elementen en geeft de gebruiker hierover feedback.

Het geïmplementeerde prototype van de applicatie is geëvalueerd door middel van een gebruikerstest. De focus van deze test ligt op het gebruiken van de applicatie om bewegingen op te nemen, opnames te herbekijken en spellen te spelen, gebruik makende van opgenomen bewegingen. De test maakt duidelijk dat interactie met de verschillende grafische elementen niet altijd natuurlijk aanvoelt. Zo is het bijvoorbeeld wel duidelijk dat de gebruiker aan een koord kan trekken ter hoogte van het handvat, maar niet dat het nodig is om de hand te sluiten om zo het effect van het vastgrijpen na te bootsen. Er is echter beperkte toelichting nodig om er zelf achter te komen hoe het mogelijk is om met de grafische elementen te interageren. Nadat de gebruiker dit eenmaal zelf heeft gevonden, is de benodigde tijd voor dezelfde interactie herleid tot een minimum. Om dit initiële leerproces te versnellen, is het mogelijk om de gebruiker een korte uitleg aan te bieden over manier waarop er interactie mogelijk is met de verschillende elementen. Verder toont de kinesist interesse in het toepassen van de applicatie in de praktijk en ziet hij eveneens andere mogelijkheden die deze applicatie biedt, zoals het gebruik voor gebarenherkenning of een interface voor deze applicatie die gericht is op het gebruik door kinderen.

Het resultaat van deze thesis is een applicatie die toelaat om bewegingen op te nemen en deze te gebruiken om met een bestaand programma of spel te interageren. De applicatie werkt onafhankelijk van zowel de gekozen bewegingen als de toepassing die zij aanstuurt met deze bewegingen. Dit laat de kinesist toe om voor de patiënt relevante bewegingen op te nemen. Opnemen gebeurt via interacties met grafische elementen die middels feedforward communiceren hoe deze interactie kan plaatsvinden. Er is een leerproces tijdens de initiële interactie met de interface, maar door het gebruiken van elementen die in het dagelijks leven kunnen voorkomen is het mogelijk om na enkele pogingen zelf te achterhalen hoe deze interactie mogelijk is. Door een minimum aan verschillende interactieve elementen te gebruiken, is het mogelijk voor de gebruiker om na het opnemen van een beweging vertrouwd te zijn met deze elementen en is het mogelijk om de applicatie autonoom en zonder verder toelichting te gebruiken.

# Contents

# List of Figures

# Chapter 1

# Introduction

There are many different reasons for requiring physical therapy. They vary from recovering from a car crash to being born with physical disabilities. The goal of these therapy sessions is respectively to make sure that the patient can recover as much as possible from his injuries or to train the muscles, preventing their situation from deteriorating.

Especially for children, the physical exercises performed during therapy can be demotivating. Combining these exercises with playing computer games leads to an increased willingness to continue with the exercises. However, the classic approach related to games that incorporate physical exercises is not economically sustainable.

The purpose of this thesis is to present a more sustainable solution to this problem in a way that offers more flexibility to the therapist, both in terms of the exercises that need to be done by the patients and the games they can control and play by performing the exercises.

## 1.1 Discussion of the problem

Patients often see exercises as a part of physical therapy as being fatiguing, monotonous and tedious. This problem is even more prominent with young patients and can quickly demotivate them, especially when the exercises are uncomfortable or painful to perform.

Computer games already exist that can support physical therapies. For instance, there are games that run on Microsoft's XBox console and accept user input through the Kinect 3D camera, or Nintendo's Wii console that use a controller with accelerometers. Additionally, there are also games that can run on a computer, using any combination of sensors for user input, and are specifically made for use by physical therapists and their patients. However, all these games have in common that they are very static by nature and are not often suited for the specific needs of a patient. For instance, a game that focuses on arm movement is not very effective at exercising the leg muscles of a patient. In other words, concerning the therapy, the game is meaningless for a patient that had leg surgery.

In addition, these static games have fixed input gestures the patients have to perform in order to control the action on screen. Even if the gestures perfectly fit the exercise requirements for a specific patient, the decrease in attractiveness of playing the same game over and over again is problematic and loses its long-term effect. Physical therapist Dries Lamberts has experience with children with disabilities and uses computer games as a form of exercise. The initial reaction of the children to these games is positive. They are more engaged in doing physical exercises and feel motivated while doing so. On the downside, this effect only lasts until the novelty of the game wears off. After that, the children lose interest in the game and, as such, in performing the exercises.

As games are expensive to develop, and motion-based games in particular, there is no wide variety of games that are tailor made for people with specific needs and at the same time offer varied gameplay mechanics to remain interesting over long periods of time. On the other hand, producing these kind of games is not economically sustainable. From the patient's point of view, the right type of exercises need to be supported by the game and the game itself has to be considered fun as well by the patients for them to keep invested in it.

## 1.2 Purpose of the research

This thesis focuses on an application that allows for a more flexible and dynamic solution to the above discussed problem. It lets the physical therapist choose what exercises the patient has to do and how these can be used as an input to control any computer game of choice. All of this can be done without requiring the therapist to have any programming knowledge.

The goal is to have an application the therapist can control mainly using the Kinect camera. It needs to be both simple and efficient to do, in order to minimize the setup time before a patient can start playing. It is necessary to research the type of interface that is required to meet these objectives, in addition to finding out the easiest way to interact with an on-screen application using a camera for input.

# Chapter 2

# Literature study

## 2.1 Human-computer interaction

Gesture-based interfaces gain popularity due to the advancing technology of sensors and processors (Jacob, Girouard, Hirshfield, Horn, Shaer, Solovey, and Zigelbaum, 2008). Because of this, technologies like the Kinect camera become more affordable and accessible for individuals. While interfaces supporting gesture input are referred to as natural user interfaces (NUI), this is more of a marketing term than an actual scientific designation. Natural interfaces don't feel natural to interact with just because they rely on gesture-based input. In fact, feedback and feedforward is limited to what is shown on-screen, but is an essential part of the interaction (Wensveen, Djajadiningrat, and Overbeeke, 2004). The user does not really hold the object shown on the screen to be able to interact with it naturally, so the feedback a user can get is limited. (Norman, 2010)

Notwithstanding this limitation, gestures are powerful tools for human-computer interaction (HCI). However, gestures as an input offer a wide variety of options and with increasing popularity, a wider variation of interfaces and gestures to interact with them exist. This is problematic as users need to get familiar to each gesture-based It's full potential can only be achieved when some kind of standard is developed and widely used.

- Limit setup time for the therapist
- simplify the process
- provide feedback for the therapist
- Papers/artikels over framework, natuurlijke interfaces,...
- ...

## 2.2 Similar research

- Vergelijking van opzet met andere oplossingen
- Te trekken lessen uit deze oplossingen (vb: feedback voor de kinesist is belangrijk, opstellen van het systeem mag niet veel tijd in beslag nemen,...)
- ...

# Chapter 3

# Design

The focus of the application is to simplify the setup process for the therapist, ensuring that he does not require programming experience or knowledge about the underlying system, while still providing all tools needed for the patients to play a game using gesture input.

The developed application can roughly be split up into three major parts. Firstly, the graphical user interface (GUI) allows the therapist to interact with the application, providing him with feedback and feedforward on inputting exercises for the patients. Secondly, gesture recognition is done as part of machine learning using support vector machines (SVM). Thirdly, all other back-end software connects the first two parts and provides a structure in which all data is managed and stored.

## 3.1 Description of the application

The application uses Microsoft's Kinect 2.0 camera as an input device. The principle is that games can be played by first recording gestures and then performing them. In order to set up the application, there are a number of steps to be taken and guidelines to keep in mind.

### 3.1.1 Principle

When playing a computer game the conventional way, a person can interact with the game by pressing a keyboard key. Pressing the key then results in an action on the screen (see figure 3.1).

KU LEUVEN

**Figure 3.1:** *Overview of the interaction between a person and a computer game under conventional use*

The developed application can be seen as an additional element between a person and pressing a keyboard key for controlling a game. In short, the application allows a person to remotely press a key by performing an exercise chosen by the physical therapist (see figure 3.2).

**Figure 3.2:** *Overview of interaction between a person and a computer game using the developed application*

More in detail, the physical therapist comes up with exercises that fit the needs of a patient. He uses the Kinect camera to interact with the application via a GUI. Next, the therapist lets the application record what exercises need to be done by the patient. Using all of the recorded exercises, an SVM model is created, which is used to predict what exercise is performed by the patient while playing a game.

The therapist assigns a keyboard button to each of the exercises. This means that when the patient mimics one of the therapist's exercises, a keyboard button is pressed. To start playing a game, the therapist can choose any game or let the patient choose his preferred game. This can be any type of computer game and includes, but is not limited to: browser games, games that need to be downloaded and installed, pre-installed games that come with the operating system,... If this application is running in the background while a computer game is opened, performing an exercise indirectly presses the button that is linked to the exercise. By doing this, the patient can interact with the game.

By mapping exercises to keyboard buttons, a vast amount of existing games can be played using gesture-based input. It is however limited to button presses. Pointing with the cursor like when using a mouse is not supported by the application. The reason is that every gesture performed is seen as one single action. The gesture of pointing to a specific spot on the screen could overlap with one of the exercises the therapist wants to use. This results in undesirable behavior and negatively impact the gaming experience for the patient. A way to solve this problem is to predefine a specific gesture that activates the *pointing mode*. In this mode, it is only possible to point to something on the screen or switch back to the *gesture mode*. However, this gives more responsibility to the therapist, who has to remember what that specific mode changing gesture is and that he cannot input that gesture as an exercise for the patient. In addition, defining a preset gesture means that not every patient has the ability to perform it, as some may not be able to move an arm or a leg, for instance. Since, from an end-user point-of-view, this setup is confusing and undoes the application's elegance of simplicity, only games can be played that require only button presses as an input.

### 3.1.2 Setup

Before the application can be used, the user has to download and install the necessary drivers in order to use the Kinect 2.0 camera. This is only required once per device it is used on. To connect the Kinect 2.0 camera to a computer, an additional adapter is needed so it can be connected using a USB 3.0 port (see figure 3.3). The use of the Kinect camera requires a socket.

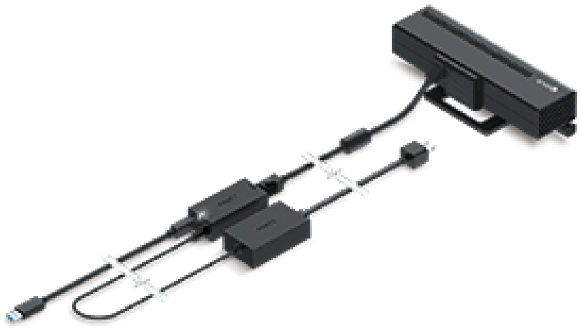

**Figure 3.3:** *The Kinect 2.0 camera connected to the adapter*

The camera has to be placed horizontally in order to function correctly. The distance between the user and the camera may vary between 1.5 and 4 meters, depending on the height of the user and the size of the used screen (see figure 3.4). The camera can be tilted upwards or downwards. The chosen angle depends on the height of the camera. Both can be chosen freely by the user, but the user has to ensure that he is fully visible at the position he stands. The GUI allows him to verify this. A possible configuration that works well is to have to Kinect camera pointing straight ahead on a table with a height of approximately 0.8 meters.

**Figure 3.4:** *Setup for the Kinect camera*

Before starting the application, it is recommended to put the executable file in a directory that does not need to change of the course of multiple uses. The reason is that a subdirectory is created in the same directory, containing all data files of saved exercises. As long as this data folder and the executable are in the same directory, all previously recorded exercises are loaded when starting the application. If they get separated, the application still can be used, but it appears as if there are no saved exercises and it creates a new data folder in the directory the executable is stored.

## 3.2 Graphical user interface

- Procesbeschrijving, opties tussen verschillende types van interfaces (mime/music conductor)
- Experimentele manier om tot prototype te komen door gesprek met kinesist
- Beschrijving van de voorgestelde oplossing, beelden van de GUI, bespreking
- Klassendiagramma voor GUI
- . . .

A large part of this thesis was focused on the design and implementation of a user-friendly interface that can be controlled via input from the kinect 3D camera. The reasoning behind this is that the user does not want to run back and forth from the standing position in front of the kinect and the keyboard. Many interfaces have been design for use with the kinect but they are generally only used to get the game and are therefore made up of simple button and scrollbars to select your option. In this section the process of developing the user-interface and the eventual implementation is discussed.

### 3.2.1 Prototypes

The goal with this part of the thesis was to find a new way of interacting with a user-interface using the kinect 3D-camera that feels natural to the average user, this despite **NORMAN's** claims that the term natural user-interfaces, that is often used to describe interfaces controlled with 3D-cameras, are not natural because of **REASONS**. This means that we will try to avoid using standard WIMP (Window, icon, menu, pointer) elements such as pushbuttons and scrollbars. A brainstorm session with the eventual user, the physical therapist, would give us fresh idea's on how the user will expect to interact with a user interface using his body or sound. The expectation was that 2 distinct observation would be made: how the user would wish to interact with the user-interface and how they would expect the flow of the program.
COMMENT: is WIMP de juiste term in het vorige stuk? COMMENT: hadden wij toen al dat visuele stappen plan? ik denk het niet he

In our process during this session the idea was to keep the user from being influenced by our idea's or by obvious solutions.To start we would only explain the basic concept of our application and the abilities of the kinect 3D-camera. Then the subject was asked how he would imagine every expect of the application, a few questions from the script were: what information would you expect to see on the screen, how would you start a recording, what would you want to see during recording, etc.

To find out whether such a brainstorm session would yield any viable results brainstorm sessions with our relatives were conducted first. An important remark to make here is that we tried to do this with people who are not from any software related area's so they had little foreknowledge about how user-interfaces are designed apart from their own experience with using programs. The 2 iterations were performed with four subjects, the first test was with two males and the second was with two females, of both genders there was one person around 25 years old and one around 55 years old. Each subject was interviewed completely separate to minimize the influence they had on each others ideas.

The sessions were very taxing for the test subjects, the concept we tried to introduce was too abstract and foreign to them. This causes them to be shy with proposals, even after being ensured that there are no bad ideas. It seemed to give them the feeling that they are be not smart enough to make good proposals. As a result they fell back on proposing familiar UI elements or concepts such as pushbuttons or voice-control. Getting locked into a mindset that we were designing games to play with the kinect was also a minor issue, thereby answering the questions with that in mind, which was not the point. After 2 iterations with minor changes between them, we decided not to subject the physical therapist to this kind of questioning. Setting up a meeting would waste to much of his precious time with little gain to us. Worse, it might instill the physical therapist with a feeling of dread for our next meeting which might reduce his level of co-operation.

COMMENT: hier mag we een of andere splitsing komen die duidelijk een onderscheid maakt tussen dees 2 delen

Instead we decided to brainstorm with each other and our supervisor Prof. Geurts to develop a few paper prototypes that we can eventually show to the physical therapist to see which one seems the most user-friendly. The two mayor interaction patterns that came out of this brainstorm session we call hints and mime. In the rest of this chapter the abbreviation UI is used to say user interface and a UI element is a button, scrollbar or any other element that the user can interact with. When discussing the different concepts we often refer to a movement by the user or position that is taken that activates a function in the program as a sign.

The hints pattern implies that the screen is largely devoid of UI elements and actions are performed when the user does a certain movement or takes a certain position, for example, the user forms an cross with his arms to delete a recording. The lack of feed forward that this implies means the user should learn all of the signs at the start of application, putting a large strain on the user's cognitive ability which is not ideal. Alternatively, all possible action could be displayed on the screen around the user with a depiction of the sign. Though this would put less strain on the user's cognitive abilities, he would still have to focus on these pictures and try to figure out what sign they depict. But what if the sign is a movement? It is impossible to depict is statically so how can you make this clear without distracting the user with constant moving pictures. A more significant problem might be the fact that position and movements might be recognized by accident, an undo movement could partly solve this but you also need a redo button to compensate for any accidental undo-activations.COMMENT: iets over het perongeluk activeren bij on aandachtig zijn. The only real benefits to using the hints pattern is that activating an action can be done from anywhere and it could be faster once the user has mastered the signs, though this depends on the ease of performing the signs and the delay between a sign and the corresponding action in the program.COMMENT: make the signs so that the link between the sign and the action is similar to the meaning it has in everyday life, culture dependency .

The mime pattern, as the name suggests, is the act of mimicking an action that is needed to act on everyday objects such as opening a door. But where an mime artist does this without any indication of that object being there, here an image of ,for example, the door would be displayed on the screen. The power of this pattern is that the user should immediately recognize the UI element, know how to interact with it. To achieve the best effect, the UI elements that are chosen should need signs that the user has done frequently already, so that doing them feel natural and familiar. An extra user friendly characteristic would be if the sign performs a function that is similar the real life such as going to a different screen when opening the door. **Wensveen** already said that for this to work correctly feedback is crucial, the reaction of the UI element must match the expectations of the user otherwise he will be confused and the flow of the program is interrupted. Disadvantages is that you cannot activate every action from anywhere and it might be slower because of that. COMMENT: Allows the use of affordances COMMENT: look in notes HCI for extra idea's from the litarute study

Our opinion is **(and that of other?)** is that one should not mix the 2 patterns because the user will expect to interact with everything on the screen when using the mime pattern if one action is activated using the hints pattern the user will forget and be confused if there is no indication of this and if there is indication the user will likely try to interact with the indicator of the sign as a button which will not have the desired effect. Combination with regular WIMP UI elements, such as pushbutton or scrollbars, is possible**(is it possible?)** for both patterns but it combines better with the mime pattern because it requires the user to point at a specific area in the UI which would detract from the idea of being able to do every action everywhere in the hints pattern. To sum up in the mime pattern the behavior of the UI elements are simple and obvious but possibly more inefficient, like pulling a handle on a slot machine to spin the slots. While the hints pattern

### 3.2.2 Description of the interface

### 3.2.3 Software

**Figure 3.5:** *The class diagram of the application, focusing on the GUI*

## 3.3 Back-end software

The focus of the application is reflected by the structure of the code. The class diagram shows a model-view pattern do make a distinction between the graphical interface of the application and the back-end (see figure 3.6).

**Figure 3.6:** *The class diagram of the application, focusing on the back-end*

The Kinect camera has the ability to identify and measure the position relative to the camera of 25 points of a person, for instance: left elbow, right elbow, head, center of the spine, left wrist,. . . All of these points are referred to as `Joint`s and can be accessed using the libraries that come as part of the Kinect SDK installation. Each `Joint` has an $x$, $y$ and $z$ coordinate, so it is unambiguously defined in space.

When the Kinect measures all of the `Joint`s at one specific point in time, these 25 `Joint`s together form one frame. This can be seen as a single picture of a person taken by the camera. Analogous to a movie, which is nothing more than a quick succession of pictures, an instance of `Gesture` collects all `Frame`s, which all together, contain information about how the person moved over a period of time.

In order to improve the application's ability to recognize an exercise, each of the exercises must be trained more than once by the physical therapist. For each exercise that is trained, an instance of `Gesture` is created. The `Gesture`s that contain data about different executions of the same exercise are grouped into a `GestureClass`. In other words, one `GestureClass` object contains all trainings of the same exercise.

After setting up a `Project` object, it contains all data that is needed for playing a game using the application. It has a `map` that maps a label for each exercise to a `GestureClass` and the actions linked to that `GestureClass`. An `Action` contains information about the keyboard key that needs to be pressed and if that key should be held down or be quickly pressed and released. The `Model` class forms the core of the application. It controls the flow of the program and contains all important objects, such as the `Project` and the `GestureClass`es.

`SVMInterface` takes all gesture data and converts it to a format that is accepted by the LibSVM library. This is an existing library that provides a C++ support vector machine implementation used for gesture recognition.

To prevent having to train all exercises again each time the application is started, all necessary data is saved into files in the data folder using the `Filewriter` implementation. This includes the data of the `Gesture`s, `GestureClass`es, `Project` and the computed SVM model. The `Filereader` is responsible for reading data from the saved data files when the application is started.

## 3.4 Gesture recognition

### 3.4.1 Support vector machine

In order to provide enough flexibility concerning the type of exercises, SVM is used. SVM supports supervised machine learning and its use encompasses two modes: train and predict.

A model can be trained with a given set of data and a label to classify the gesture. The amount of numbers in one data set is referred to as the number of features. If it contains $n$ features, the entire data set can be seen as a single point in a $n$-dimensional space. It is possible to train the same gesture more than once. In that case, the same label is used to indicate that the given data set is related to the same gesture. In other words, all trainings of the same gesture are linked to the same label. All of these trained gestures with the same label are seemingly similar, but actually contain variations due to noise during the measurement of the gesture or a slightly varying execution of the gesture.

After training all required gestures, a SVM model is created. Given a data set of a gesture, this model can predict which of the trained gestures has the biggest resemblance to the given data set. As a result, the label of the most similar trained gesture is returned. This also explains the necessity of having multiple trainings recorded for each gesture. Errors in a single training due to noisy measurements of the Kinect camera can lead to wrong predictions. Having multiple trainings minimizes the influence noise has on the prediction and keeps into account that the user executes all gestures with slight variations. As a result, the model can predict more accurately which gesture is performed.

However, there are some things to keep in mind when applying this strategy. Firstly, while inputting multiple repeats of the same gesture helps with predicting the gesture after a model is created, it takes more time for the therapist to do this. Secondly, when trying to predict a gesture, the generated SVM model always returns the label of the most similar gesture, even if they are not related at all.

These problems are tackled as part of the approach to using SVM to predict gestures.

### 3.4.2 Approach

As stated in section 3.3, a gesture consists of multiple frames. Each frame contains 25 joints and each joint consists of an $x$, $y$ and $z$ component. This results in a total of 75 features that are being considered for each frame.

Two approaches are considered when it comes down to learning how to recognize gestures. By directly comparing these approaches, it is easier to identify their advantages and disadvantages.

The first approach is to add a time stamp to each frame as an extra feature, which indicates the time relative to the first frame of the gesture. This results in having 76 features per frame. If performing a certain gesture takes about $t$ seconds and is captured at a rate of $f$ frames per second, this amounts to 76 $tf$ features. This additional feature can be used to make sure that the gesture isn't just similar in space, but also in time. For a 3-second gesture recorded at 30 frames per second, which is the highest sampling speed of the Kinect camera, this amounts to 6840 features for a single training of the gesture. In other words, the number of features of one training depends on the duration of the gesture. The entire gesture is classified as a single gesture. During prediction, a gesture with the same number of features can be input to verify if that gesture matches any of the trained gestures.

There are several problems with this first approach. If the number of features of the gesture used for predicting does not match the number of features of the training gestures, the prediction is not accurate and should be discarded. Discarding is necessary as SVM always returns the label of a gesture, even if the predicted gesture is completely unrelated to any of the trained gestures. This poses a problem as different gestures can have a different duration. Even different trainings of the same gesture can take for instance 3 seconds and 3.1 seconds, implying that the recordings have a different number of frames and thus a different number of features. A possible solution is to recalculate the entire gesture and use interpolation to convert the set of frames to a new set with a known, fixed size and preferably with equidistant time stamps.

Furthermore, not just the trainings of one gesture, but all of the gestures need to have the same number of frames. The gesture to predict is not known beforehand and can only be predicted accurately if the number of features for both the predicted gesture and trained gesture are equal. As a result, it is required that all gestures have an equal amount of features. This means that short gestures need to be mathematically extended with additional frames to match the size of longer gestures. Another side effect is that all gestures need to have the same duration, not just the same amount of features. This limits the therapist in choosing exactly what exercises he wants the patient to perform. Also, predicting a gesture can only be as fast as the trained gesture with the longest duration. More in particular, assume the longest gesture is 5 seconds in length. It then follows that it takes about 5 seconds to predict any gesture. This also means that patients that control the game can only perform one action every 5 seconds. As the games to be played are unknown beforehand, a slow reaction time of the application can render some games unplayable. As such, this approach is not viable.

The second approach is to split up one gesture into $n$ smaller gestures and classify each of them differently, assigning a different label to each part of the gesture. Assume that a gesture is split up into 4 smaller parts so that each part contains approximately an equal amount of frames. Consider a 4-second gesture sampled at 30 frames per second. The complete gesture consists of 120 frames. By splitting it up into 4 parts, each part contains 30 frames. All first 30 frames are labeled with the same label, for instance 1. The next 30 frames receive a label 2, and so on. To put it differently, the considered gesture is split up into 4 postures with each having 30 slightly varying trainings. In contrast to the first approach, as described above, prediction does not happen for an entire gesture, but for separate postures. The condition for having executed the entire gesture is that each of the postures are predicted correctly and in the right order. To put it differently, $n$ *pictures* are taken of the gesture and if at some point during prediction all pictures are executed in the right order, the application acknowledges the execution of the gesture.

AFBEELDING NODIG MET EXTRA UITLEG IN ALINEA HIERBOVEN

The biggest advantage of this approach is the flexibility of it. Each posture corresponds to a single frame, which contains 75 features. This eliminates the need of having gestures with the same length or having to modify training data to have gestures with an equal number of features. It is even possible to not just link a gesture, but also a posture to a keyboard button, which allows the physical therapist to choose the exercises that fit the needs of a patient the best. Additionally, the application reacts immediately to an executed gesture, not after a fixed amount of time. This allows for a wider variety of games being playable using this application.

Another advantage is that it solves the issue where SVM always returns the label of a predicted gesture, even when the patient is not doing anything. A gesture is only recognized when all parts of it are executed. In other words, gestures are not recognized involuntarily and, as such, actions like button presses that influence the game are not executed by accident. This also means that no neutral or *wrong* gestures are required to link a certain gesture to no in-game action.

The disadvantage of this second approach is that there is no strict requirement for the entire gesture to be executed in about the same time as the gesture recorded during training. If a gesture is executed faster during prediction compared to during training, it is recognized as the same gesture being executed. However, it is possible to solve this timing issue to some extent when the gesture during predicting is performed slower. The gesture can be ignored if more than a certain amount of time passes. This time threshold takes the gesture with the longest duration into account and is chosen higher than this in order to allow all gestures to be executed and successfully recognized.

# Chapter 4

# Implementation

The application is developed in C++ using Microsoft's Visual Studio IDE and can run on computers with a Windows operating system. On an implementation level, platform-specific features are used for saving and loading data files and running the GUI. On a hardware level, the Kinect 2.0 camera requires the installation of a driver in order to function properly. This comes as part of the *Kinect for Windows SDK 2.0*. Microsoft's download page states the system requirements for using the Kinect 2.0 camera. These requirements include: a 64-bit processor, dual-core 3.1 GHz or faster processor, USB 3.0, 4 GB of RAM or more, graphics card that supports DirectX 11, Windows 8 or 8.1, Windows Embedded 8 or Windows 10.

## 4.1 Graphical user interface

## 4.2 Back-end software

After initialization of variables like the instances responsible for the user interface, the model and the communication with the Kinect camera, the program locks into an infinite loop while the application is running. This main loop consists of three processes: fetching new data from the Kinect camera, analyzing this data in order to use it for recording or predicting gestures and updating the GUI with relevant changes.

It is important that none of these three processes block the continuous flow of the main loop. The update rate of the interface is only as fast as the rate with which the main loop is executed. If it is slowed down with a blocking or long-running loop, the GUI appears to stutter of freeze and no new data is collected from the Kinect camera.

The Kinect camera can only provide new data as fast as 30 times per second. If no new data is available at the moment the main loop requests the data, for instance when the main loop is executed faster than 30 times per second, the program skips this process and continues with the other two processes.

To control the flow of the program without reverting to blocking loops, software flags are used. These flags keep track of the current state of the program. By evaluating the state of the flags, it is possible to indicate if the program is predicting a gesture, recording a gesture or animating specific graphical elements of the GUI.

## 4.3 Gesture recognition

It is possible to split up the implementation of gesture recognition in two parts: the recording of a gesture by the physical therapist and the prediction of a gesture executed by a patient.

### 4.3.1 Recording a gesture

```
void Model::recordGesture(Frame & frame) {
  if (!initialized) {
    //Set the neutral frame on the first call
    frameNeutral.setFrame(frame);
    initialized = true;
  }

  //Add the frame to the buffer
  framesBuffer.push_back(frame);

  if (!startedMoving) {
    if (! frame.equals(frameNeutral)) {

      startedMoving = true;
      framesBuffer.clear();
    }
    else if (framesBuffer.size() > NOT_MOVING_FRAME_DELAY) {
      addRecordedGesture();
    }
    else {
      return;
    }
  }

  if (framesBuffer.size() > NOT_MOVING_FRAME_DELAY &&
      framesBuffer.back().equals(framesBuffer.at(framesBuffer.size() -
      NOT_MOVING_FRAME_DELAY))) {
    addRecordedGesture();
  }
}
```

**Code snippet 4.1:** method to record a gesture

### 4.3.2 Predicting a gesture

Code snippet 4.2 shows a piece of code.

```
bool Model::isGestureExecuted(int checkLabel, int posInBuffer, int recursiveCounter, int
    badCounter) {
  //The label exists and is linked to a posture.
  if (activeProject->containsLabel(checkLabel) &&
      activeProject->getGestureClass(checkLabel)->getGestures().front()->isPosture())
    return true;

  //Done enough recursive checks to confirm the gesture has been executed.
  if (recursiveCounter >= NB_OF_LABEL_DIVISIONS)
    return true;

  int nextLabelToCheck = checkLabel - 1;
  for (int i = posInBuffer; i >= 0; i--) {
    if (labelsBuffer.at(i) == nextLabelToCheck)
      return isGestureExecuted(nextLabelToCheck, i, recursiveCounter + 1, 0);
  }

  //If this point is reached, a label that is one less than the given
```

```
  //  label cannot be found in the buffer.
  //The gesture may still have been executed, so keep checking for the
  //  next one if the badCounter is not too high.
  if (badCounter > 0)
    return false;

  return isGestureExecuted(nextLabelToCheck, posInBuffer, recursiveCounter + 1, badCounter +
      1);
}
```

**Code snippet 4.2:** method to verify if a gesture with given label is executed

# Chapter 5

# Results

User tests make it possible to verify if the design meets the expectations of the target group and offer a way to discover unexpected problems that are hard to track by the developers themselves. Test persons have a fresh look on the application and can provide useful insights in the way they interpret visual elements of the user interface.

In addition, the technical evaluation considers the effect of the implementation, ensuring that the user is not hindered by performance issues or similar problems.

## 5.1 User test

The graphical user interface is designed for use by physical therapists. As such, acquiring feedback from therapists is essential in developing an interface that is both useful and easy to use. The process of the user test is described, followed by an overview of the most important results of the evaluation itself and an interpretation of these results.

### 5.1.1 Description of the test

A user test is conducted with physical therapist Dries Lamberts at Windekind Leuven, an organization that focuses on the education of children with disabilities and offers a full program to assist them with their specific difficulties. For this user test, the Kinect camera is connected to a computer that runs the application and positioned on a desk at hip height. The user indicates that he is familiar with using the Kinect camera and motion-based games.

A short introduction on the purpose of the application is given to the test person for reference. After that, he is presented with several tasks and is asked to apply the think-aloud protocol in order to obtain as much information as possible about the interaction with the application and the thoughts of the user while doing so.

In preparation of the user test, gestures are pre-recorded that both serve the purpose of testing the robustness of the application as well as having a fallback plan in case difficulties arise during the test.

One task is to watch the pre-recorded gestures and control a game called Space Invaders using the pre-recorded gestures. This is a game that has an avatar move left or right and shoot bullets. As such, four gestures are pre-recorded: one gesture that allows the avatar to move to the left, one to move to the right, one to shoot bullets and a neutral gesture that is linked to no keyboard key. At this point, it is explained to the user why a neutral gesture is required. The goal of this task is twofold. Firstly, the task is used to verify if the test person is able to watch replays of the pre-recorded gestures without any guidance and to learn what gestures are pre-recorded without additional information. After this is done, it is explained to the user how the gestures and keyboard keys are linked together. Secondly, it makes it possible to confirm if a correct prediction of a gesture execution does not depend on the person performing the gesture, as problems could arise due to the person recording the gestures and the person playing a game have a very different physique.

Another task is to play the game Sokoban Geek. This puts the gamer in control of an on-screen avatar that can move up, down, left or right. Walking into boulders allows the player to move them in order to solve a puzzle. The game is shown to the test person and after trying the game, he decides how many gestures are required to play the game. Then, he comes up with different gestures for each input and records three trainings for each of them. The test person is asked to delete a recorded gesture and record it again. After recording all gestures, he tries to play the game. The goal of this task is to check how the test person reacts to the recording elements of the interface and if it is clear how to interact with them without assistance.

With permission of the test person, during the entire test, the computer screen is recorded, as well as everything said, for analysis purposes. The system usability score (SUS) allows to obtain a numeral indication of the usability of the application based on ten questions. At the end of the user test, the user is asked to fill out a questionnaire with these ten questions.

### 5.1.2 Evaluation

For the first task, the user notices the recorded gestures in a list on the right-hand side of the screen. By trying to move its avatar's hand to align with the screenshots of the recordings, he notices that it is possible to scroll through the list of recordings. He also notices the delete option next to the recordings list and indicates that he is anxious about deleting one of the recorded gestures.

Watching a replay of the recorded gestures initially is not clear to the user. He tries to drag one of the gestures to the square on the left hand-side of the screen. He notices that this is not possible, but makes several attempts do this. After that, he indicates that he can't find it and looks at the interface without trying to interact with anything. To replay a recorded gesture, the user can point with the avatar's hand to the recorded gesture in the green area of the scroll bar. At that point, a replay of that gesture is displayed automatically. The replay stops while scrolling through the list or when not pointing at the gesture in the green area. The user did this several times during this task subconsciously, without noticing that a replay started playing at the left side of the screen. The user required assistance to complete this task. He was able to do this after knowing that the gesture in the green area is displayed at the left-hand side square.

After watching replays of the recorded gestures, the user is able to replicate in real life what gestures are recorded and displayed on-screen. When the Space Invaders game is started, the user has no problem interacting with the game and does not require any assistance. He is able to control the in-game avatar subtly and meet the game objectives. After asking for his experience with the controls, he indicates that the controls are responsive and that he does not experience any lag.

For the second task, the user gets the chance to see what the Sokoban Geek game looks like and how it is played. After that, he states that he needs five gestures to play the game. Four gestures are linked to a directional keyboard key and one gesture that serves as the neutral gesture. To record a gesture, it is required that the user points the hand of the on-screen avatar to the grip of the pulling cord, makes a fist, and move his fist down to start recording. This translates to pulling the displayed cord. Trying to start recording the first gesture, the user pulls the cord without any problems. The recording screen appears and he starts performing a gesture when the interface indicates that recording has started. He stands still when he is done with the gesture and the recording stops automatically.

Upon being asked to delete the gesture he just recorded, he needs no assistance to push the gesture object to the right into a box indicating that the gesture can be deleted. He indicates that his earlier anxiousness of deleting a gesture by accident is unnecessary, as it requires some effort to perform this deleting action and it is hard to do it by accident.

When trying to record a gesture the second time, the user is unaware that he is required to close his hand into a fist to pull the cord. He tries pointing at the grip and moving his hand down and indicates that isn't sure why recording does not start this time. Only after several tries, he tries closing his hand to pull the cord, activating the recording screen. He states that he did not know that closing the hand was required and that he did that by accident to record the first gesture. He indicates that he has experience using the Kinect camera for other games and that these games never required to close the hand, so he was not sure that the Kinect could make the distinction between open and closed hand.

The user notices clearly that the recording of a gesture stops when he stands still. Because he was not aware of this, one gesture was not recorded the intended way, but the user was able to delete this gesture and record a new one without assistance.

When trying to watch a replay of a recorded gesture, he is not sure if he needs to point at the gesture with open or closed hand. Although both cases function in the same way, the user states that using a closed hand feels like the application responds better.

After the user records all gestures, he is able to play the game. However, the gesture that is linked to the left arrow key does not respond. The user is asking if the key is linked correctly to the gesture and thinks this is the problem. He misses feedback about what gesture is linked to what key. After verifying for the user that the key is linked correctly, we present the user with the question of how he would handle this problem without any assistance. He suggested to delete the recorded gestures and record them again. After this, the user is able to play the game without further problems.

The entire user test approximately took 45 minutes. After the test, the user asked because of his own experience with disabled children if very small gestures are supported by the application and if the application can be used for other purposes than playing games. An example he thought of was using gestures to type words using sign language. He also added that he was glad that use of the space bar is supported by the application, as he has experience with similar projects that do not support the space bar, while it is one of the most used keys in computer games. However, a lot of games the children are interested in require the ability to point with a cursor on-screen. This is a feature he would appreciate. Finally, he stated that he is very interested in using the application, allowing disabled children to play games and exercise while doing so.

### 5.1.3 Interpretation

The physical therapist proved to understand the concept of the application. This proves that it is possible for persons without programming knowledge to use the application for recording gestures and controlling computer games. He also understands that the use of the application is not limited to games and can also be used for other computer applications that require keyboard key input.

A few tries are required to get familiar with how to start recording and how to end it. In particular, without prior knowledge or experience, more feedforward is required to communicate that the hand needs to be closed when pulling the cord and that the user needs to stand still to stop recording. However, after a few tries, it is possible to record multiple gestures in succession without any of them not being recorded as intended.

Replaying recorded gestures requires the user to align his on-screen hand with the gesture inside the green area of the scroll bar. There is no form of feedforward indicating that performing this action results in the gesture being replayed. The feature is intended for the user to be discovered while scrolling through the gestures or while trying to delete a gesture. As such, this can be confusing to the user.

Both the actions of recording gestures and replaying recorded gestures are performed faster after having experienced this firsthand. The provided feedforward is too subtle to immediately make it clear what actions are expected. To improve this, it is either possible to provide more explicit feedforward, or to introduce the user to the most important functions using a tutorial.

## 5.2 Technical evaluation

The application is developed with the Microsoft Visual Studio integrated development environment (IDE) in C++ and can run on Windows operating systems. This limitation is due to the use of the IDE's visual interface editor and Direct2D, as well as the feature of saving and loading data files. The application is tested to run on both Windows 8.1 and Windows 10 machines.

All of the required files for running the application, including resource images, take up approximately 20 MB of space. Saving all exercises needed for playing a game takes up approximately an additional 0.6 MB of space. This depends on the number of exercises recorded, the number of trainings done for each exercise and the duration of each exercise.

# Chapter 6

# Discussion

## 6.1 Reflection on the results

## 6.2 Reflection on the process

## 6.3 Future work

To further develop the application,

- Reflectie behaalde resultaat
- Reflectie proces
- Voorstellen voor verbetering (efficiëntie,. . . )
- Voorstellen voor toekomstige projecten (vb: focus op besturing door kinderen ipv kinesisten)
- . . .

# Chapter 7

# Conclusion

- Samenvatting probleemstelling, implementatie en resultaten
- ...

# Bibliography


R. J. K. Jacob, A. Girouard, L. M. Hirshfield, M. S. Horn, O. Shaer, E. T. Solovey, and J. Zigelbaum. Reality-based interaction: a framework for post-wimp interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI 08(April):201 – 210, 2008.

D. A. Norman. Natural user interfaces are not natural. *Interactions*, 17(3):6 – 10, 2010.

S. A. G. Wensveen, J. P. Djajadiningrat, and C. J. Overbeeke. Interaction frogger: a design framework to couple action and function through feedback and feedforward. *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*, DIS 04(August):177 – 184, 2004.

FACULTY OF ENGINEERING TECHNOLOGY
CAMPUS GROUP T LEUVEN
Andreas Vesaliusstraat 13
3000 LEUVEN, Belgium
tel. + 32 16 30 10 30
fax + 32 16 30 10 40
fet.groupt @kuleuven.be
www.fet.kuleuven.be