Fachgebiet Medizintechnik

im Institut für Mikrotechnik und Medizintechnik
an der Technischen Universität Berlin

Professor Dr.-Ing. Marc Kraft

Bachelor's Thesis

Development of a movement game with inertial sensors for children with cerebral palsy aged 2-4 years

10.02.2023

Submitted by **Rebekka Mirbach** (371703)
Supervised by **M.Sc. Christina Mittag**

Dovestraße 6, 10587 Berlin, Telefon 03031423388

Bachelorarbeit

Technische Universität Berlin

Fakultät V Verkehrs- und Maschinensysteme
Institut für Maschinenkonstruktion und Systemtechnik
Fachgebiet Medizintechnik
Prof. Dr.-Ing. Marc Kraft

Für
Frau Rebekka Mirbach
Matr.-Nr. 371703

Thema: Entwicklung eines Bewegungsspieles mit Inertialsensoren für Kinder mit Zerebralparese im Alter von 2-4 Jahren

Die bimanuelle Nutzung der Hände ist eine grundlegende Fähigkeit für die Teilhabe am Alltag. Von basalen Fertigkeiten, wie der Aufnahme von Nahrung, bis hin zu komplexen motorischen Kompetenzen, wie dem Schrifterwerb, ist eine Nutzung der Handfunktion die Voraussetzung für eine altersgerechte Entwicklung und Selbstständigkeit bei Heranwachsenden. Die Erkrankung der unilateralen Zerebralparese kann zu Einschränkungen und Störungsmustern einer Seite führen. Kinder aller Altersgruppen profitieren von intensiven Trainingsmaßnahmen mit der betroffenen Hand, um die motorische Funktion (wieder)herzustellen.

Im Rahmen des Forschungsprojektes „multimodales sensorgestütztes Hand- und Armfunktionstraining für Kinder" (SHArKi) wurden erste Bewegungsspiele für Kinder im Grundschulalter entwickelt. Im Rahmen dieser Arbeit soll ein Konzept für ein Bewegungsspiel für Kleinkinder entwickelt und umgesetzt werden. Das Spiel soll durch Pronation und Supination gesteuert werden.

Die Arbeit gliedert sich in folgende Aufgabenteile:

- Einarbeitung in die Thematik durch eine Recherche zu folgenden Schwerpunkten:
  - Zerebralparese
  - unimanuelle sowie bimanuelle Handtherapie
  - Eulerwinkel
- Anforderungen an Game Engins und Auswahl einer geeigneten
- Spielkonzept mit Motivationselementen
- Softwarearchitektur und Umsetzung des Spiels nach V-Modell
- Testung des Spiels
- Diskussion und kritische Einordnung der Ergebnisse

Die Betreuung der Arbeit erfolgt durch M. Sc. Christina Mittag. Nach Abschluss der Arbeit sind ein Vortrag zu halten sowie ein Abstract in deutscher und englischer Sprache anzufertigen.

Berlin, 31.10.2022

Prof. Dr.-Ing. M. Kraft

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 10.02.2023

Rebekka Mirbach

Ausblick

Diese Arbeit befasst sich mit der Entwicklung einer Bewegungsspielsoftware im Rehabilitationskontext für Kinder mit Zerebralparese. In dem System werden die Bewegungen Supination und Pronation trainiert. Die Bewegungssteuerung erfolgt dabei über eine Intertialmesseinheit, den Movesense Sensor. Ziel des Systems ist es, die Kinder durch Motivationselemente zu regelmäßigen Trainingseinheiten zu ermuntern und damit ihre motorischen Fähigkeiten zu fördern. Es wird der Versuch verfolgt, das System von zu Hause zugänglich zu gestalten, indem es als Handyapplikation ausgelegt wird. Für die Softwareentwicklung wurde der Ansatz des V-Modells angewendet. Aufgrund der spezifischen Anwender*innengruppe wurden zunächst die Rahmenbedingungen im Detail betrachtet. Eine geeignete Game Engine wurde ausgewählt und ein Spielkonzept, das die Therapieanforderungen erfüllt, ausgearbeitet. Die dabei angewendeten Methoden sind die Nutzwertanalyse und die Untersuchung des Therapiekontextes anhand einer Hospitation, sowie einem Expertinnengespräch. Das Spielkonzept wurde in der gewählten Game Engine implementiert. Es wurde eine Android Applikation entwickelt, in der die Bewegungssteuerung des Spielablaufs mit dem Movesense Sensor realisiert wird, der am Ende eines Stabes befestigt ist. Als Interventionsmethode wurde die bimanuelle Therapie ausgewählt. Das System enthält drei motorische und drei kognitive Level, um Kinder mit unterschiedlich starken Einschränkungen angemessen zu fordern. Es besteht die Möglichkeit, dass das entwickelte Spiel zu anspruchsvoll für Kinder mit Zerebralparese ist, um als Trainingssystem genutzt werden zu können. Zudem muss die Effektivität von Heimtrainingssystemen für die Verbesserung der motorischen Fähigkeiten der Anwender*innen kritisch betrachtet werden. In weiteren Schritten könnte das System in User-Tests und Durchführbarkeitsstudien validiert werden.

Abstract

In this work, a movement game software was developed for children with cerebral palsy, to train the movements supination and pronation. The motion control is realized by an inertial measurement unit, the Movesense Sensor. The system aims to encourage the children to train on regularly and thereby promote their motor skills. It was attempted to make the system accessible from home by designing it as a mobile phone application. The software was developed using the V-model's approach. For that, the framework conditions in the rehabilitation context were considered in detail due to the very specific user group. A suitable game engine was selected and the game concept fulfilling the therapy's requirements has been established. The methods applied are the benefit analysis and the exploration of the therapy context conducted through a hospitation and an expert discussion. The game concept was implemented in the chosen game engine. An Android application was developed in which the movement control of the gameplay is regulated by the Movesense sensor that is attached to a bamboo stick. The intervention method chosen for the game is bimanual therapy. The system contains three motor and three cognitive levels to appropriately challenge children with varying degrees of impairment. There is a risk, that the developed game is too demanding for children with cerebral palsy, depending on the severity of their condition. In addition, the effectiveness of home training systems for improving users' motor skills must be further critically analysed. In future, the system could be validated in user tests and feasibility studies.

Contents

List of Figures

List of Tables

List of Abbreviations

BLE Bluetooth low energy

CIMT Constraint-induced movement therapy

CP Cerebral palsy

CS Coordinate system

GFA Game flow android

GFC Game flow computer

HABIT Hand arm bimanual intensive therapy

IMU Inertial measurement unit

MACS Manual ability classification system

MP Movesense-Plugin

MPA Movesense-Plugin final application

MT Motion tracking

NPCA Non-progressive congential ataxia

OS Overall system

ROM Range of motion

SG Serious game

SGE Serious game engine

SMA Scene management android

SMC Scene management computer

UI User-interface

USCP Unilateral spastic cerebral palsy

V Value

WF Weighting factor

WV Weighted value

1 Introduction

This work is a continuation of the project SHArKi,[1] realized by the medical technology department of Technische Universität Berlin in cooperation with the companies GETEMED GmbH and nova motum Services & Consulting GmbH. In course of the project, a multimodal sensor-based hand and arm function training was developed for children with the non-progressive motor disorder cerebral palsy. Aiming to improve the motor function by regular exercising, games were designed to train motions like flexion and extension or adduction and abduction [50].
As complement to the completed SHArKi project a new rehabilitation system using a different motion sensor and focusing on another movement is developed. The new system is equipped with the Movesense sensor [77] by the finish company Suunto [80]. The movements to be trained in the game are supination and pronation. For the successful implementation of a game, a suitable game engine is first being sought. In the SHArKi project, the game engine GDevelop was used to develop and implement the games. This platform is contrasted with the two game engines Unity by Unity Technologies and Unreal Engine by Epic Games. Requirements are established and the engines are then compared in a benefit analysis. The evaluation of the comparison leads to the choice of the most fitting engine.
The target audience for the game are children aged $2-4$. Ideas for game concepts involving supinator and pronator muscles are collected and the question if the gaming involves both hands (bimanual hand therapy) or only the affected one (unimanual hand therapy) is addressed. Based on a job shadowing in a physiotherapy practice and the experience from the practice's therapists, a concept and the intervention method is chosen. A motivating game concept is then further designed, evaluated, and eventually implemented in the selected game engine. Due to different severity levels in the player's disease and the resulting individual movement restrictions, the game will contain multiple motoric levels. To be mentally demanding, several cognitive levels will be implemented as well. The role as a parent of a child with cerebral palsy is commonly experienced as isolating and stressful [51]. To ensure their ability to help their affected child develop to the best of their ability, their mental health is crucial to protect [51]. That is why, additionally, the aim is pursued to develop the game as home training. The system being home-based aims to keep the child independently occupied with a meaningful task while giving parents a time out.
The goal of this scientific work is the design and implementation of a serious game for children with Cerebral palsy aged $2-4$ years. Therefore the theoretical foundations are firstly examined in chapter 2. In chapter 3 the employed methods and the conceptual design are introduced. Following, in chapter 4 the results are

[1] `https://www.medtech.tu-berlin.de/menue/forschung/projektseiten/sharki/`, last accessed 12.09.2022

displayed, which are then discussed in chapter 5. Finally the conclusion is drawn in chapter 6.

The questions addressed in the research's concerns are summarised in the following:

1. Which game engine is suitable for the implementation of the rehabilitation game?
2. Should both hands be trained in the game (bimanual practice) or does it make more sense to focus on the affected side (unimanual practice)?
3. How shall a rehabilitation game for children aged $2 - 4$ years, in which the movements supination and pronation are trained, be designed to be interesting to play, challenging enough for an effective practice, but at the same time not over-demanding?
4. How can the existing system be modified to make it accessible from home and thereby integrable into the children's daily life?

2 State of the Art

In this section, the disorder cerebral palsy is presented in subsection 2.1. The subsection is further divided and the following topics are introduced: MACS, a classification system for cerebral palsy patients 2.1.1, the movements supination and pronation 2.1.2, that are to be trained in the game, and therapy methods for motor impairments using one or two hands 2.1.3. In the second subsection, serious games for motor rehabilitation are reviewed 2.2. In the third subsection 2.3, user-centred system design is described. Following, motion measurement is explained 2.4. In this context, inertial measurement units 2.4.1 and rotations 2.4.2 are presented. Concluding, three game engines are introduced 2.5: GDevelop 2.5.1, Unity 2.5.2 and Unreal Engine 2.5.3.

2.1 Cerebral Palsy

The non-progressive motor disorder cerebral palsy (CP) is a neurodevelopmental condition, starting in early childhood and enduring throughout the lifespan [13]. The accurate meaning of the term and the best way to classify the physically disabling condition is being discussed until today [12]. CP is part of a group of permanent disorders that cause movement and postural impairments, hereby leading to limitations in daily activities [12]. The deficits occur due to the impairment of cerebral structures in the pre- or perinatal period, before the completion of brain maturation [58]. Possible aetiologies of the disorder are believed to be pre- or perinatal oxygen deprivation, cerebral haemorrhage, brain malformations, metabolic or genetic disorders, maternal infections or infections of the central nervous system, toxic damage, or traumatic brain injury [58][47], but frequently the cause remains unknown [58]. Manifestations and symptoms of the disorder are determined by the lesion's time of occurrence in brain development and its localization [58]. Besides sensory and motor deficits, CP is associated with cognitive impairment, behavioural problems as well as communicational and motor disabilities [45]. Participation issues for patients with CP include self-care like getting dressed, personal hygiene, eating, productivity concerning school, household management or work and leisure activities like socializing [18]. In childhood, CP is considered to be the most common cause of severe permanent physical disability [45]. Since the individual cases differ greatly, classification systems attempt to form groups. The use of multiple classification systems enables to accurately describe individual CP cases and allows optimal function-focused therapy. Most commonly, classification systems are based on the pattern of motor impairment, the distribution of symptoms and/or the severity of the disability.
The classification based on the different types of movement disorders divides patients with CP into three phenotypical categories: spasticity, ataxia, and dyskinesia [3].

75% of children diagnosed with CP suffer from spastic forms [6]. According to the european classification system, spastic cases display a minimum of two of the following characteristics: They exhibit abnormal body posture and/or motion patterns, and/or their tone is increased, and/or their pathological reflexes are increased [3].

CP presenting with ataxia, also termed non-progressive congenital ataxia (NPCA), affects around 10% of children with CP [60]. The origin of the word comes from the Greek etymology and it means "lack of order" [60]. In ataxis Patients present with the inability to coordinate their movements [9]. The performed motions are executed with abnormal force, inaccuracy and out of sync [9][3]. The patients' muscle strength is not affected but the execution is disturbed [60].

Dyskinesia affects approximately 15% of CP cases [6]. It is characterized as involuntary, uncontrolled, recurring, and occasionally stereotyped movements as well as abnormal body posture and/or motion patterns [3].

The distribution classification differentiates patients according to the region of the body involved. Unilateral vs. bilateral involvement can further be differentiated by the number of limbs involved, as visualized in figure 1. The darker shades of green imply the colored region's greater impairment [35].

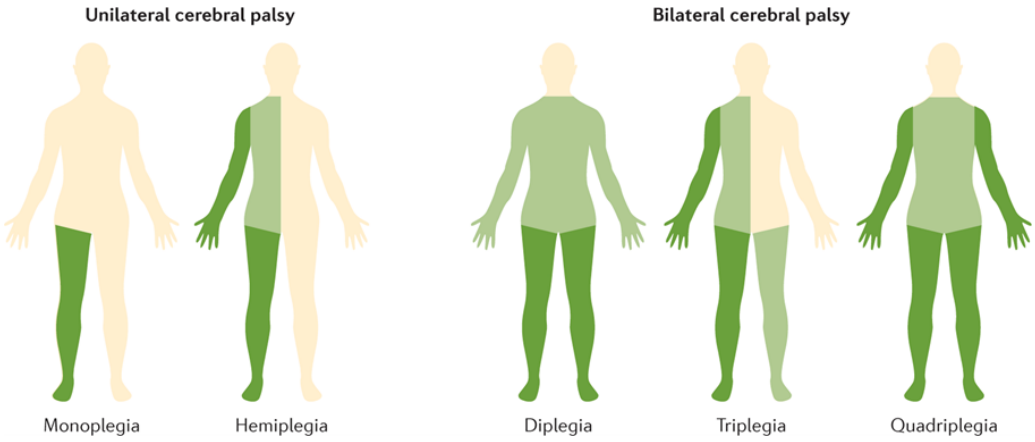


Figure 1: Topography Classification of CP [35]

The diagnosis of bilateral spastic CP implies that the affected limbs are located on both body sides, whilst for unilateral spastic CP the localisation is limited to one side of the body [3].

CP can also be classified by the severity of symptoms. One classification system is the Manual Ability Classification System (MACS). The MACS is used to assess the ability of children with CP to manually handle objects during activities of everyday life. It categorizes the patients by functional performance [37] and is further explained below in subsection 2.1.1.

Unilateral spastic cerebral palsy (USCP), is the most common presentation of CP in general (30% of all patients with CP) [60]. With one healthy body side, these

children tend to have difficulties in bimanual coordination, since they have the urge to only use their unaffected hand spontaneously and to neglect the affected limb [60]. This work focuses on the USCP subgroup.

2.1.1 Manual Ability Classification System

The Manual Ability Classification System (MACS) takes daily activities under consideration to classify how well children are able to handle objects with their hands. Not only the affected hand, but the collaborative use of both hands together is rated. Children's capabilities to manually perform are differentiated into five different levels. The classification is based on quantity and quality of performance as well as independent execution or the need for assistance [7]. Since younger children naturally need more support and supervision and handle objects differently than teenagers, two ranking systems exist: MACS (age $4-18$) and Mini-MACS (age $1-4$). They rate the ability of both hands in age-appropriate everyday-life situations. The second one is of interest for this work. The Mini-MACS leaflet is listed in the appendix A and its classification system is summarized in table 1.

Table 1: Mini Manual Ability Classification System (Mini-MACS)[7]

Mini Manual Ability Classification System	
Level I	Handles objects easily and successfully
Level II	Handles most objects but with somewhat reduced quality and/or speed of achievement
Level III	Handles objects with difficulty; needs help to prepare and/or modify activities
Level IV	Handles a limited selection of easily managed objects in adapted situations
Level V	Does not handle objects and has severely limited ability to perform even simple actions

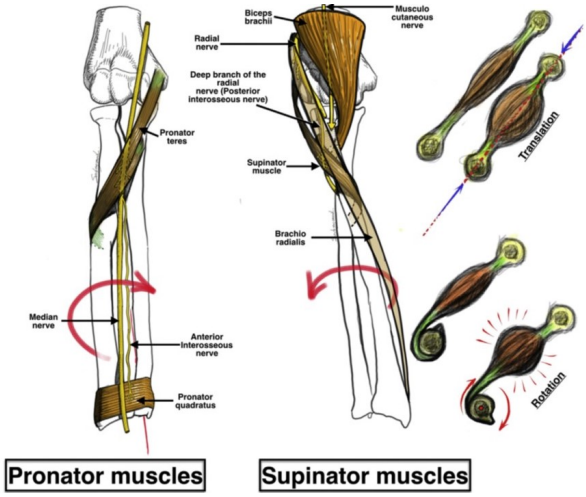
For developing the rehabilitation system for patients with CP aged $2-4$, the varying degrees of disease severity and the multifaceted clinical pictures of the target group must be considered. Therefore, this work is in the following restricted to USCP cases, classified in level one to three. Since the movement game aims to train supination and pronation, explained in subsection 2.1.2, the therapy methods introduced in subsection 2.1.3 are limited to the upper half of the body.

2.1.2 Supination and Pronation

Supination and pronation are significant for everyday activities like drinking, hairbrushing, face washing, getting dressed, and more [33]. Supination is the rota-

tional movement of the hand, pointing the palm upwards [38]. From patient's perspective for the right arm the forearm rotates clockwise and for the left arm counterclockwise [33]. In the pronation movement of the hand, the palm is pointing downwards [38], with the forearm rotating clockwise for the left arm and counterclockwise for the right arm [33]. In figure 2 the movements are visualized in subfigure 2a and the stressed muscles are shown in subfigure 2b. The muscles involved in supination are the supinator [33], biceps brachii and brachioradialis [38]. For pronation the pronator teres [33] and pronator quadratus muscle [38] are used.

(a) Supination and pronation of the right hand [33]



(b) Pronator and supinator muscles [38]

(c) Typical posture of child suffering from USCP [5]

Figure 2: Supination and pronation in context of CP

In patients with CP, the supinator muscle is weakened [20]. In comparison, the two muscles providing pronation are considered too strong [20]. To execute reach-to-grasp and gross motor tasks, children with USCP, lack supination of the forearm as well as elbow extension in comparison to healthy children [17]. One approach to strengthen elbow extension could be weakening the biceps muscle. However, for the improvement of functional range, caution is required, since the biceps muscle is the strongest supinator of the forearm when the elbow is flexed [20]. Instead, the focus of treatment should be set on changing the ratio of the flexion

and biceps' supination moment towards a more pronounced supination function [20]. It is clinically known that for children with CP the execution of an active supination movement is more difficult, leading to an extended movement time and adding to the tasks' complexity [17]. With forward flexion of the trunk the lack of supination is tempted to be compensated [11][17], hereby resulting in the typical posture of children suffering from spastic CP, as shown in figure 2c.

2.1.3 Unimanual and Bimanual Hand Therapy

Therapists and families are provided the opportunity for safer and more effective interventions, thanks to the expansion of the CP evidence data base in the last decade [24]. Nevertheless, some interventions are based on out-dated neurological theories about the disorder. This is one of the reasons for 6% of the intervention methods being considered ineffective [24]. Effective therapy requires regular outcome measurements and knowledge of recent neuroscientific and pharmacological findings [24]. For all CP cases, individualized, interdisciplinary medical care throughout the lifespan is of great importance and can lead to more independence, participation and a higher life quality for the patients [47]. The medical care consists of physiotherapy, orthopaedics and neurology, with individual needs varying widely [47]. Yet physiotherapy has a central role in all CP treatment plans [26]. To provide optimal conditions for the children, smooth cooperation between parents, patients and specialists from the areas mentioned above are essential [47]. A diagnosis at a young age comes with the advantage of being able to intervene early, to select a fitting rehabilitation technique and to improve the rehabilitation process [47]. The choice of an appropriate form of therapy for a child lays an important foundation. Therapy methods to treat patients with CP's upper limbs can be categorized into different therapeutic interventions.
For this work, the therapy method is selected between unimanual and bimanual therapy. A central question from a clinical point of view is by which of the two forms the overcoming of the developmental disorder is easier to achieve [21]. Both therapy forms enable constant improvement of the hand function in unilateral children [21][24] and are considered to be effective interventions for motor impairments [51]. An application of unimanual therapy is referred to as constraint-induced movement therapy (CIMT)[21]. Its goal is to achieve an intensive use of the affected upper extremity [37]. Learned nonuse describes the phenomenon when a disabled limb is not being used despite the presence of sufficient motor innervation [8]. To overcome this phenomenon intensive training and restraint of the unaffected arm is intended to increase the movement of the affected hand in a goal-directed manner [8]. To accomplish this, the less affected arm is restrained while performing structured tasks [37][60]. For the constraint aids such aids like plaster bandages, gloves, slings, mitts [60] or casts [24] are applied. Children may find this method difficult, since by applying the impaired limb alone, CIMT

requires a high level of initiative and commitment [60]. Furthermore the children need to deal with frustrating experiences, caused by regular difficulties and failures [60].
Bimanual training provides equal effectiveness to CIMT [24]. It is also referred to as hand arm bimanual intensive therapy (HABIT) [60]. In this treatment form, both hands are used together to complete a task [37][24]. Instead of constraints, the training incorporates game variations that encourage bimanual exercises [60]. Thereby the focus is set on the affected limb's movements which include stabilization, cooperation and handling activities [60]. Possible tasks in HABIT therapies include playing cards, ball games, building blocks, and crafting [61]. In bimanual activities, unilateral cerebral palsy patients mainly use the affected limb as an assisting hand [11]. Nevertheless, HABIT strengthens the cooperation of both hands by using the upper limbs and better functioning is achieved since "two hands are better than one" [16]. Frequently, HABIT is used after CIMT, to enable the integration of the upper limb unilateral strengthening into ambidextrous abilities [37].
Improving bimanual and occupational performance and increasing movement efficiency is supported by either of the training methods [31]. However, trials comparing the dosage of therapy use in home practice between unimanual therapy and bimanual therapy showed the achievement of higher dose in HABIT [28]. This may indicate that HABIT is easier to implement as home practice than unimanual training [28].

2.2 Serious Games for Motor Rehabilitation

Serious Games (SGs) are video games with another primary purpose than entertainment [39]. In the healthcare field this purpose is reaching a goal like treatment, recovery, and rehabilitation in an engaging and entertaining way [39]. The games address cognitive processes as attention and motivation with the aim to maintain or improve the player's motor movements [27]. Since therapy usually consists of repetitive and intensive activities, a loss of interest and of concentration in the patient may be the result, having a negative impact on the effectiveness of the therapy program [27]. SGs are beneficial in putting the user's attention focus on the game and not on therapy [27]. Furthermore, they give the patients the opportunity to carry out training sessions independently from home without a therapist's supervision [15]. A voluntary increase in therapy dosage is achievable by considering positive psychology and motivational components in addition to motor learning in the sessions [23]. When the games create a fun and engaging environment they bear the opportunity to motivate the child to exercise regularly [15]. More active and intense training may eventually contribute to improvements in functional outcomes [15][21][61]. The use of SGs for patients with CP as a supplement to traditional therapy could therefore be of interest [26].

According to Novak's systematic review, they may be considered as an adjunctive intervention for children with CP [51]. Attractive prospects are offered by technological advancement and lower costs on the patients and clinicians side[26]. In addition, research benefits from the fact that patient movement data can be recorded, measured and evaluated in the virtual world [15]. For patients with CP, limitations in self-mobility as well as pain are associated with a lower quality of life [10]. Providing access to simple treatment like motor rehabilitation therefore holds the potential to improve the patients' and their families' lives [59]. Its aim is to promote greater participation in society by reducing impairments and increasing functional ability [15].
The usage of virtual reality-, web- or computer-based interventions comes with the potential to improve muscle strength and to thereby increase independence within the MACS level [37]. Nevertheless, the concrete efficiency of SGs needs to be further researched [26]. In a trial, the efficiency of a custom SG system was tested with the outcome of an improvement in the children's test group hand function [41]. A validation of the results has not yet been carried out [41]. The newest review on the use of serious gaming to improve sensorimotor function [59] could not give clear answers on the effectiveness since treatment outcomes are impacted by the heterogeneous nature of the disorder and the variety of different study designs. However, the conclusion was met that serious gaming in addition to a conventional treatment may be a useful tool [59]. The interventions are non-invasive, don't lead to a deterioration and potentially improve compliance [59].
In order to be promising, the player must not be bored but must be given the feeling of having fun while learning [46]. A motivating structure generates voluntary participation, long playing times and a high re-playability [23]. This is why design frameworks are introduced in the following.

2.3 User Centered System Design

To generate a high level of patient motivation in a SG, the question of interaction design is crucial [27]. Regardless of the therapy's approach being unimanual or bimanual, in a practice or at home, the interventions design, targeting and functional orientation adapted to the patient can have a wide influence on the child's well-being [19]. This perception concerns feelings of both functioning and participating, as well as physical health [19]. To ensure that the child does not get bored, a variety of games and difficulty levels is strongly desirable [59]. If an overarching storyline exists, children may push to achieve goals they are working towards in a medium or long term [59]. The success of a SG is dependant on the usage of multiple design principles [27]. Three frameworks were included in the design process, listing different key factors with similar messages. All three frameworks share the view that choice and interactivity are crucial for successful games [46] [27] [23]. The user should be given the option to express personal taste by allowing

selection of themes in the implementation [27]. Another common key factor is the necessity to define precise and easy to understand instructions to ensure correct participation. This is termed as development paradigm [27], rules [46], or clear goals and mechanics [23]. Additionally, game mechanics provide motivational factors that aim to enable interactive learning [46]. Motivation is also triggered by rewarding experiences [23][46]. Furthermore, feedback is a key factor, as the user's achievements are thereby acknowledged [23][46][27]. Moreover, it serves to inform the user on his or her current state in motor learning [27][23]. Further factors are demanding difficulty levels and the opportunity to play the game with multiple players to thereby socialize [23].
One approach implements the issues adaptability, monitoring and clinical evaluation [27]. Adapting the game to specific needs includes providing motor levels that depend on the user's skills and giving the therapist or parent the option to set a maximum playing time [27]. To grant the therapist the opportunity to monitor the user and his progress a dataset is useful, in which information about the user's performance are stored [27].

2.4 Motion Measurement

Motion tracking (MT), also known as motion capturing, is used to record and analyse human movements [52]. It finds application in various fields, including the entertainment industry, sports, robotics, and medicine [56]. With motion capturing the measurement of fine-grained biological movement is enabled, being increasingly accurate, accessible, and child-friendly [52]. In the analysis of the provided data, done online, or offline, motion features, or kinematics are distilled [52]. Angular velocity and acceleration of the patient are simultaneously obtained [53]. The data of interest are the differential properties of joints, velocity and acceleration [52]. Different branch types to track motions with varying properties are on the market, e.g. marker-based systems, markerless methods and inertial motion tracking [52]. For this work, only the movement of the upper limb is of interest, especially of the wrists. To measure kinematic data of human bodies at any location, portable inertial measurement units (IMUs) can be used [53]. In the SHArKi project, Xsens Mtw[2] IMUs were utilized [50]. The sensor provided for this work is the Movesense Sensor HR+[3], designed and produced by the company Suunto in Finland. The sensor is shortly introduced in the following subsection 2.4.1. More information as the technical highlights can be reviewed in the data sheet in the appendix D. The sensor's motion tracking is evaluated in real time and is to be converted in such a way that it can serve as motion control. For this purpose, the motion that results from supination and pronation of the hands, the rotation, is described in subsection 2.4.2.

[2] `https://www.xsens.com/products/mtw-awinda`, last accessed: 20.12.2022
[3] `https://www.movesense.com/movesense-active/`, last accessed: 20.12.2022

2.4.1 Inertial Measurement Units

Inertial sensors is the term for sensors that measure their own motion [36]. In doing so they also measure the rigid body's motion to which they are attached [36]. They are represented by accelerometers, measuring linear acceleration, by gyroscopes, measuring angular velocity, and by magnetometers, sensing the local magnetic north [36][52]. The arrangement with three orthogonal gyroscopes and three orthogonal accelerometers mutually aligned to each other is called IMU [36]. The small and lightweight units can be stored in objects or worn on the body [36]. They are used to estimate motion variables like orientation, velocity and position [62]. IMUs have benefits like low cost and energy consumption, a reasonably accurate orientation frame, portability [56] [45], the detection of rapid movements and the usability in tight spaces and large capture areas [52]. Their disadvantages are the length of the data's postprocessing [56] and that it is necessary to assess the position by integrating angular velocity or accelerations [52]. Doing so, a cumulative error follows, termed an inherent drift [52], since the data is affected by high noise levels and time-varying biases by surrounding unwanted magnetic flux [32]. These errors can be partially corrected when using analysis methods [52]. The estimation of the attitude is done by comparing the reference vectors with the gravity and local magnetic field [32]. Problems occur when ferromagnetic objects or electronic devices disturb the local magnetic field [32]. Using a sensor fusion algorithm, a smooth and distortion-free estimate of orientation can be obtained by processing the data [32]. However, employing algorithms can still lead to errors in the orientation components [32].

The orientation components of the attitude estimation are termed yaw, pitch, and roll [32]. They represent the rotations around the three cartesian coordinate axes [2]. The rotation around the x-axis is referred to as roll, around the y-axis as pitch and around the z-axis as yaw [32]. The components are visualized by [56] in figure 3.

Subsequently, the applied Movesense sensor is reviewed. The battery powered, small sized sensor is a nine-axis motion sensor, containing accelerometer, gyroscope, and magnetometer. Like other IMUs it generates time-series data from the accelerometer, gyroscope, and magnetometer around the three axes [56]. Real-time streaming is possible with wireless connectivity via Bluetooth Low Energy (BLE) [78]. Other units alternatively have embedded wireless transmitters or use SD cards for data recording [36]. In the medical field, IMU have been used for e.g. neurological diseases and to measure the range of motion (ROM) in children with CP [45]. However, further studies are needed for confirming the clinical and predictive value of IMU measurements [45].

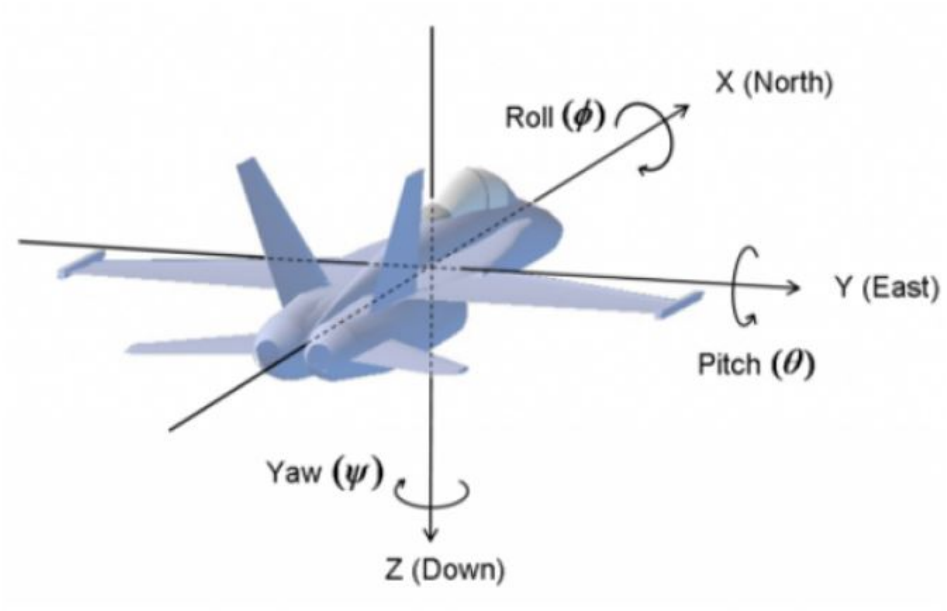


Figure 3: Rotations around the axes [56]

2.4.2 Rotation by Quaternions and Euler angles

The three-dimensional space of orientations is parametrized by Euler angles or quaternions [2]. Three Euler angles $(\theta_1, \theta_2, \theta_3)$ are required to describe orientations in $\mathbb{R}^3$. The first angle θ_1 represents the rotation angle about the x-axis, the second angle θ_2 represents the rotation angle about the y-axis, and the third angle θ_3 represents the rotation angle about the z-axis [2].
The movements supination and pronation require the rotation of a body around an axis in three-dimensional space. The concept rotation describes a rigid body moving around a defined axis [43]. Both are introduced in the following and their advantages and disadvantages are described.
Euler discovered in his theorems that every rotation has an axis [1]. It is referred to as Euler axis [43]. The rotation matrix encoded in Euler angles is commonly used to denote a rotation about an axis through its origin in the three-dimensional space ($\mathbb{R}^3$) [57]. The matrix is 3×3 orthogonal and has the determinant 1 [57]. Euler angles are considered as rotating x-, y- and z-axes. While the first rotation affects only itself, each subsequent rotation affects the previous one [63], meaning that they are not commutative [43]. Between the axes exists a hierarchy [63]. The rotation order yxz with the rotation matrix $R_z(\theta_3), R_x(\theta_1), R_y(\theta_2)$ is shown in figure 4 [63].

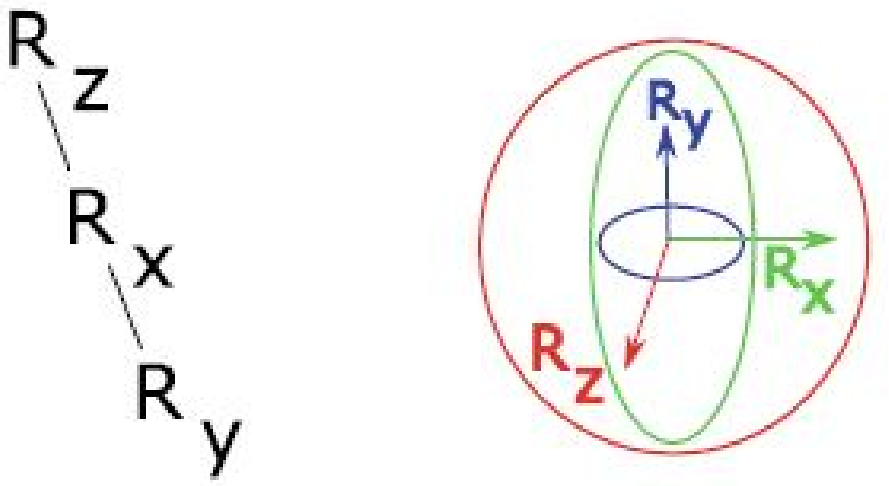


Figure 4: Hierarchy between Euler angles [63]

If two axes of the Euler angles become parallel, gimbal lock occurs [57]. The object in the three dimensional space is locked in one direction. It can only rotate within two dimensions and has lost a degree of freedom [94]. This problem can be avoided by using quaternions for the rotation's parametrisation instead [43]. Whilst matrices are used to represent transformations like translation, scaling, projection, and shearing, quaternions are commonly used to represent rotations and orientations [2]. In comparison, to compose two rotations in $\mathbb{R}^3$ by product of two matrices, requires 27 multiplications and 18 additions , while two quaternions only require 16 multiplications and 12 additions [57].

Complex numbers represent geometric processes in the two-dimensional space [64]. William Rowan Hamilton tried to find three-dimensional numbers serving equally well for these processes in the three-dimensional space [64]. In 1843, he discovered four-dimensional numbers, following laws of calculation [64]. These numbers are the quaternions, denoted with the symbol $\mathbb{H}$ [64]. They are hyper-complex numbers that can be expressed as a four-membered vector or as a sum [43]. Quaternion $q \in \mathbb{H}$ has a scalar part $s \in \mathbb{R}$ and a vector part v= $(x, y, z) \in \mathbb{R}$ [2]. It is by definition equal to [2]:

$$q \equiv [s, \mathrm{v}] \equiv [s, (x, z, z)] \equiv s + ix + jy + kz. \tag{1}$$

One number describes the scaling's size, one number describes the to be rotated number of degrees and two numbers describe the rotation plane of the vector [2]. To represent any general three-dimensional rotation θ about axis n unit quaternions are used [2]. Unit quaternions $q \in \mathbb{H}$ are quaternions of length

$\|q\| = 1$ [2]. Their rotation is described with a unit vector n of length $\|n\| = 1$ and the angle rotation θ . It counts [2]:

$$q = \left[cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right) n \right]. \tag{2}$$

To apply a rotation of θ around the axis n to a point r $= (x, y, z) \in \mathbb{R}^3$, a new quaternion p is constructed, having r as its imaginary part ($p = [0, r] \in \mathbb{H}$) [2]. The rotated point r' is then the imaginary part of the quaternion p' defined by [2]:

$$p' = qpq^{-1}. \tag{3}$$

Subsequently, the two coordinate systems relevant for this work are introduced and visualised in figure 5. The first one is the respective Movesense sensor's coordinate system, displayed in figure 5a. The second one is the world space's coordinate system, visualized in figure 5b. The left-handed system was chosen because it corresponds to the coordinate system of the engine that will be used later on (3.3.3).

(a) Movesense sensor's CS (b) World space's CS

Figure 5: Coordinate systems (CS)

2.5 Game Engines

The success of a SG depends on the quality of the SG's engine: the serious game engine (SGE)[29]. The use of a proposed framework helps developers identify the best choice of SGE [34]. On account of the large variety of engines available, a preliminary choice was made to only consider the following tools:

1. GDevelop, as it was used for the development and implementation of games in the preceding SHArKi project,
2. Unity by Unity Technologies, as it offers ready-to-use elements, intuitive tools, and tutorials and is therefore popular among newcomers in game development [49],
3. Unreal Engine by Epic Games, as it is especially well-known for its high-class graphics.

The logos of the three engines are shown in figure 6.

(a) GDevelop [81] (b) Unreal Engine [83] (c) Unity [82]

Figure 6: Logo's from the three engine's twitter accounts

Below the three engines to be compared are briefly introduced: GDevelop 2.5.1, Unity 2.5.2 and Unreal Engine 2.5.3.

2.5.1 GDevelop

GDevelop is a no-code, open-source, and easy to use game-making app [72]. It allows for the creation of only 2D content. The content can be exported in one click and runs on all platforms except for consoles. The focus of the engine is put on the simplicity of usability. Instead of code, events are used to express the logic of the game. That way, time on learning how to use a software is saved and can be spent on developing a good game [75].

2.5.2 Unity

According to the company, in 2021 more than 50% of games across devices (mobile, PC, and console) were made with Unity [97]. On the real-time development platform, creations of 2D, 3D, Virtual Reality (VR) [87] and Augmented Reality (AR) content can be realized. Unity creations can be run on more than 20 platforms, including Linux. The Unity Asset Store offers a variety of 962 packages and 79994 assets moderated by Unity Technologies. With a 5−Star-Rating-System, they are rated by more than 85000 customers. The asset can be filtered for the developer's specific needs, including the categories, the pricing, the ratings, and the platforms. For a 3D game on a Linux Platform it offers 1600 assets free of charge [85]. The variety of free items give the developer the opportunity to realize different designs. Unity offers a Support Site where the developer can search their knowledge base. If the base doesn't provide an answer, it is possible to submit a request, use the platform's forums or answer pages. Further help can be found via online search engines. Due to the big community of Unity developers, many blogs and forums concerning the developing process are available. The engine is free of charge for non commercial goals. The Personal license allows developers to develop commercial games for free if the earnings from the game are less than 100000$ in revenue. If they're higher, Unity Plus, Unity Pro, or Unity Enterprise can be purchased. The coding language used is C# [98].

2.5.3 Unreal Engine

Unreal Engine by Epic Games is a real-time 3D creation tool, claiming to be the most open and advanced of the world [99]. For game developers the software is free until the earnings of a title exceed 1000000$ [99]. The download includes access to full source C++ code, integrated tools and features. On the platform documentation, tutorials and support resources are available, as well as free content like templates, samples, and complete projects [103]. To learn how to properly use the engine, online learning content and a library of webinars is offered, as well as instructor-led trainings [104]. As a widely used engine, there is a big community equivalent to a strong support-base behind it.Unreal Engine's asset store is finely sorted but lacks free objects [101].

3 Method and Conceptual Design

In this section the software development process is described. The rehabilitation system is developed under the approach of the methodology V-model. Using this approach, the feasibility of the system is already considered during the planning phase, because individual components are designed and tested before the entire system is implemented [40]. The approach helps the developer in transforming the game concept into an applicable system. The method is introduced in general terms in subsection 3.1. Subsequently, the requirements for the overall system are set up in subsection 3.2. Then, the system's framework conditions are examined in detail. This is required due to the specific environment in rehabilitation context for a small user group of children with CP aged $2 - 4$. In order to create the framework, first the selection of a suitable SGE is conducted in subsection 3.3. Second, an in-depth investigation of the therapy context to determine a suitable game concept is laid out in subsection 3.4. Then, the implementation of the game using the V-model's approach is described in subsection 3.5. Finally, the system's requirements are verified in subsection 3.6.

3.1 V-model

The V-model is a method that commonly finds application in the field of software and system engineering [40]. System engineering is based on the principle of developing a technical system within given constraints such as time and budget [48]. The V-model is suitable for software development, because it specifies clear goals and control tests for the development process [55]. It provides a structured systematic approach to develop mechatronic and cyber-physical systems [55]. The developer is guided through the development process by the structured framework with tools that simplify the development process [55]. Furthermore, the framework helps the user orientate in the interdisciplinary development process, by linking together the respective engineering tasks of all disciplines involved in the product development [48]. Applying the approach aims to support the user in successfully developing the complex technical system [55]. The latest version of the V-model is illustrated in figure 7. The methodology is described in detail in the following.

On the left side of the "V" letter's thigh the main system is decomposed into segments and subsystems [55]. On the right thigh, they are then gradually reintegrated into the overall system [55]. Therefore, the left thigh is referred to as the development side and the right thigh as the test side. The three parallel running strands represent in blue the modelling and analysis of the system, in orange the core tasks of the system development, and in yellow the requirements engineering [55]. Throughout the development process each working step is constantly

verified and validated [55]. The strands pass through checkpoints that represent the system's progress [55]. In figure 7 they are indicated by the numbers 1 to 6. Each checkpoint contains control questions to support the developer in keeping track of the working progress [55]. Throughout the development of a successful product the questions serve as a helping tool for the structure and completeness of the development process [48].

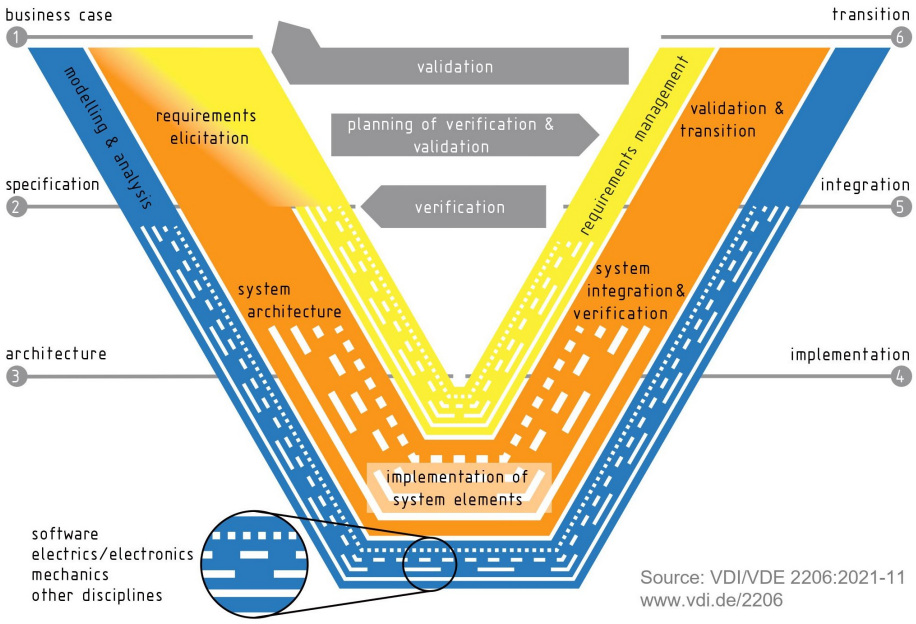


Figure 7: V-model VDI/VDE 2206 [106]

The core tasks depicted in the middle strand are processed iteratively and partially overlap [55]. They consist of five subsequent tasks: **requirements elicitation**, **system architecture**, **implementation of system elements**, **system integration and verification**, and **validation and transition** [55].

The elicitation, structuring, and analysis of requirements is essential as foundation for the system's development process [55]. Therefore, in the first core task **requirements elicitation**, system requirements are defined. Those are the conditions and capabilities elaborated on the basis of customer requirements that must be met by the system or subsystem [55]. Requirements are of functional or non-functional nature and their relevance is divided into demands and wishes [25], as further explained in subsection 3.3.1. The collection of requirements can be structured in main-feature lists, in which the requirements' characteristics are categorized into main and secondary features [55].

In the second core task, the **system architecture**, a cross-disciplinary overall solution structure of the system is developed on the basis of the specification [55]. The technical system design is prepared, which includes the structured planning of the idea and the definition of the technical layout [55]. Verifiable entities such as system components with assigned specific requirements and corresponding interface specifications are created [55].
The **implementation of system elements** is the next step based on the two preceding core tasks [55]. The developer implements the components according to the main-feature list and the architectural draft [55]. The design, dimensioning, programming and implementation of the specific system elements are therefore generated [55]. For software elements, algorithms are applied to implement functions and the corresponding code is written [55]. Furthermore, problems that occur during the fourth core task in the system integration are being solved [48].
The section **system integration and verification**'s aim is to guarantee that the integrated overall system meets the previously specified requirements [55]. It serves to merge the implemented system elements step by step into the next higher system level and ultimately into the overall system [55]. It is on the opposite side of the core task system architecture and therefore validates and verifies the technical draft. The verification process addresses the question: "Has the system been developed correctly?" [55]. The requirements are verified by measures like component tests or analysis [55]. Unit, integration, and system testing are performed according to the software testing's specifications [22]. The results of the testing are summarized in the test completion report [22]. After the completion of the level's verification, the integration into the next higher core task takes place [55].
In the highest system level **validation and transition** the fulfilment and implementation of the stakeholder's needs and requirements is examined [55]. The questions addressed are: "Was the right system built? Is the customer satisfied? Were the requirements correct? Do they represent the needs of the stakeholders?" [55]. When the last checkpoint is passed and the overall system was verified and validated, it is transmitted to the stakeholder [55].
During the whole process, side processes take place in the inner and outer strand. The inner and outer strands contain the tasks of **requirements management**, and **modelling and analysis**, and are performed parallel to the core tasks [55]. Throughout the working process it is possible that changes occur in the requirements addressed in the **requirements elicitation**. This is e.g. due to gain in knowledge or updates in the customer's wishes [55]. The constant management of the requirements is therefore inevitable. It "serves to analyse, structure, assign, and integrate changes in requirements throughout the development project" [55]. The outer strand's task **modeling and analysis** is a requirement for efficient computer-based development [55].

Every mechatronic or cyber-physical system is individual. Therefore partial steps can be omitted or processed less extensively, depending on the system's needs [55]. To develop the overall system for this work, subsystems need to be prepared first. Each individual subsystem represents an integrated system incorporated into the main system [55].
Following, the **requirement elicitation** for the overall system is performed, described in subsection 3.2. Then, the framework conditions for the software development are established using the methods of a benefit analysis and an expert discussion. In subsection 3.3 the SGE is chosen, and in subsection 3.4 the game's design is discussed. The system's software development process is described in subsection 3.5. The overall system is decomposed into three subsystems that are addressed in separate subsections. The first system is the general game flow with its further subsystems, the second one is the Android environment, and the third one is the connection and motion control by the Movesense sensor software. In the subsections 3.5.1, 3.5.2, and 3.5.3, the method is applied and accordingly simplified for the system that has to be designed.

3.2 Overall Systems's Requirements

The general requirements for the overall system are relevant when determining the framework conditions. Therefore they are already introduced here. They are the first core task of the software development process, that is later described in subsection 3.5. In the **requirement elicitation**, the system is regarded from above in the black box view. The functional system design is elaborated, which does not yet include a description of the technical implementation.
For this project, the defined requirements are derived from the stakeholders demands, from the SHArKi project's standards and from the developers ideas, that are based on literature research. It was requested by the customer that the system is designed for a specific age group, focusing on a specific movement and using a specific sensor. In addition, the developer decided to make the system accessible from home by developing it as an Android application. The home-based training systems is advantageous because it might lead to a voluntary increase in therapy dosage and grants the opportunity of practising without supervision, as already stated in subsection 2.2.
The requirements for the overall system are listed in the main-feature list in table 2. They each have a main feature, secondary feature and are either obligatory requirements (Relevance D) or optional requirements (Relevance W). The system to be developed is from now on referred to by the project name **Movement Game**.

Table 2: Main-feature list for the system Movement Game

#	Requirement	Main Feature	Secondary Feature	Relevance
1	Training the movements supination and pronation	Function	Signal	D
2	Age range of users $2-4$	Organisation	Stakeholder	D
3	Children with CP	Organisation	Stakeholder	D
4	Multiple cognitive and motoric level	Organisation	Stakeholder	D
5	Search for BLE device	Function	Signal	D
6	Data transmission via BLE device	Function	Signal	D
7	Connection setup with BLE sensor	Function	Signal	D
8	Enable home-based therapy by running the application on an Android operating system	Realization	Environment of use	W

If the Movement Game meets all criteria listed above is reviewed after the implementation of the system's subsystems in subsection 3.6. Beforehand, the framework for software development is prepared in the following two subsections and the system is implemented.

3.3 Choice of the Game Engine

In this section, a SGE from the three engines GDevelop, Unity, and Unreal Engine, introduced in subsection 2.5, is chosen. Firstly, requirements are defined to ensure optimal conditions for the development process, described in subsection 3.3.1. Secondly the requirements are evaluated in a benefit analysis, as described in subsection 3.3.2. The preparation of the requirements and the benefit analysis is realized according to Pahl/Beitz [25]. Finally, the engine with the highest total benefit value is selected, stated in subsection 3.3.3. It is later used for the software development, further examined in subsection 3.5.

3.3.1 Requirements for the Game Engine

Initially, requirements are discussed, elaborated, and defined in exchange with the customer and are then collected in requirement lists [25]. These lists are the document that is used for product specification e.g. in development departments [25]. In the listed requirements a distinction is made between demands and wishes [25]. Demands must be satisfied in all circumstances [25]. Without their fulfilment, the

proposed solution is not acceptable at all [25]. If a variant does not meet one of the demands of the requirement's list, it is therefore eliminated [25]. Wishes on the other hand should be taken into account whenever possible [25]. This may possibly be accompanied by a concession to a limited amount of additional work [25]. In the following, demands will be referred to as obligatory requirements and wishes as optional requirements. Whilst the obligatory requirements are inevitable, the optional ones might benefit the outcome's quality and lead to a more pleasant developing experience.
The requirements for the SGEs are derived from the overall system's main-feature list (2) and from the developer's preferences. At this stage the requirements of the three subsystems, the game flow 3.5.1, the Android environment 3.5.2 and the Movesense sensor's software 3.5.3 must already be considered. Main criteria are established, which are then split into further sub goals if required. The criteria, separated into obligatory and optional requirements are enumerated in the simplified requirement list 3.

Table 3: Requirement List for the SGEs

Obligatory requirements	
1	**Gameflow**
1.1	Opportunity of implementing multiple cognitive and motoric levels
2	**Android**
2.1	Working in an Android environment
3	**Movesense sensors**
3.1	Compatibility with the sensor's software
Optional requirements	
4	**Engine's usability**
4.1	Simplicity of Development
4.2	Asset Store
4.2	Community
5	**Game's graphics**
5.1	3D graphics
5.2	2D graphics

Project Name: Movement Game
Date of Creation: 27.09.2022
Date of last Modification: 08.01.2023
User: Developer (Rebekka Mirbach)

The functional main requirements concern the overall system's subsystems, introduced in subsection 3.1: the game flow, the Android environment, and the sensor's software. The engine should enable the option to build an application that runs on Android devices. Furthermore, the engine needs to be compatible with the Movesense sensor's software. It also needs to grant the opportunity to implement multiple cognitive and motoric levels for the game flow, as demanded by the stakeholder.
The first non-functional, optional requirement, sought by the developer, applies to the SGE's usability. It is divided into three sub goals, that were considered due to the following aspects. A user-friendly, easy to extend, and fast to prototype SGE is advantageous for newcomers in game development. Besides that, the development process is simplified through a well-stocked asset store and a large community by which gaming forums are updated. The second non-functional requirement concerns the quality of the graphics. The graphics can be realized in a 2D or 3D environment. Developing in a 3D environment is not obligatory but brings benefits as it increases the amount of options to design in an engaging way. The requirements are rated in the following to then evaluate the best fitting SGE in the benefit analysis 3.3.2.

3.3.2 Benefit Analysis

The benefit analysis is the analysis of the set of complex alternative actions [30]. Its purpose is to order the elements of the set with respect to a multidimensional target system [30]. This is done according to the decision maker's preferences [30]. The benefit value is the total value of the respective options [30]. The procedure goes back to Zangemeister, who first introduced it in 1970 [25]. The requirements are rated by using weighting factors. A weighting factor is a real, positive number, indicating the importance of one evaluation criterion compared to others [25]. The requirements are weighted with weighting factors in the range of zero to one [25]. The sum of the main criteria's weighting factors respective to each other and also for the main criteria's sub goals is one [25]. This is to achieve a percentage weighting of the requirements among each other [25]. For each requirement the solutions are rated. In the benefit analysis, a value spectrum from 0 to 10 is applied [25]. As the value increases, so does the suitability. The value spectrum can be reviewed in the book Pahl/Beitz [25]. Three values are given here as examples: 0 points indicate that the solution is absolutely useless [25], 5 points indicate a satisfying solution, and 10 points indicate the ideal solution [25]. The benefit of the different alternatives can be compared by performing the following calculations. The values that are determined in each case are multiplied with the weighting factor [25]. The total value is then determined by building the sum of the weighted values [25]. The benefit value indicates the engine's suitability. The benefit analysis is portrayed in table 4. The weighting factors

are referred to in the table by WF, the values by V and the weighted values by WV. The main criteria gameflow and Movesense sensors are weighted with $0,3$ points. They only have one sub goal which is therefore weighted with 1 point. Building in an Android application is weighted with $0,2$ points. The optional requirements are each weighted with $0,1$ point. For the usability, the sub goal simplicity of developing in the SGE is weighted higher than the Asset Store and the community.

Table 4: Serious Game Engine's benefit analysis

			GDevelop		Unity		Unreal	
#	Requirement	WF	V	WV	V	WV	V	WV
1	**Gameflow**	**0,3**						
1.1	Opportunity of implementing cognitive and motoric levels	1	10[73]	10	10	10	10	10
	Intermediate Value			3		3		3
2	**Android**	**0,2**						
2.1	Running on Android	1	10[72]	10	10[67]	10	10[100]	10
	Intermediate Value			2		2		2
3	**Movesense sensors**	**0,3**						
3.1	Compatibility with system's sensors	1	10	10	10	10	10	10
	Intermediate Value			3		3		3
4	**Engine's usability**	**0,1**						
4.1	Simplicity of development	$0,5$	10	5	8	4	6	3
4.2	Asset Store	$0,3$	2[74]	$0,6$	10[85]	3	6[105]	$1,8$
4.3	Community	$0,2$	4[76]	$0,8$	8	$1,6$	8[69]	$1,6$
	Intermediate Value			$0,64$		$0,86$		$0,64$
5	**Game's graphics**	**0,1**						
5.1	3D graphics	$0,7$	0	0	10[29]	7	10[29]	7
5.2	2D graphics	$0,3$	10	3	8[29]	$2,4$	0[29]	0
	Intermediate Value			$0,3$		$0,94$		$0,7$
	$\sum$	**1**	$8,94$		$9,8$		$9,1$	

In source [29], the SGE's features were rated from zero for the feature's non-presence, to five for excellent. The rating's scaling was adjusted to the benefit analysis' ranking and hence multiplied by two.

While GDevelop has no code and is therefore easy to use[4], C# is easier to learn than C++ [102]. The quality of the asset store is rated by tidiness and the availability of free objects and templates. The ratings for the community are made under consideration of its size, availability and quality of blogs, forums and tutorials. Developing in the two-dimensional environment is easier to realize than in the three-dimensional but the sub goal 3D graphics is weighted higher than the 2D graphics one. This can be justified by the outcome of a data analysis, indicating that players interact in a 3D environment in a more naturalistic way, having lower reaction times and higher correct answers than in 2D systems [54]. For better comprehension, the calculation is demonstrated for the requirement "Game's graphics" for the SGE Unity: The requirement was weighted with one tenth point (weighting factor WF $= 0,1$). The sub goal 3D graphics was weighted with seven tenth points (WF $= 0,7$) and the sub goal 2D graphics with three tenth points (WF $= 0,3$). The sum of the two sub goals's weighting factors is one, just like the sum of the main criteria weights:

$$0,7 + 0,3 = 1, \qquad 0,3 + 0,2 + 0,3 + 0,1 + 0,1 = 1.$$

Unity's 3D graphics are considered excellent (Value V $= 10$). Multiplying the value with the sub goal's weighting factor leads to the sub goal's weighted value (WV $= 10 \cdot 0,7 = 7$). The 2D graphics are considered very good (V $= 8$). Again the sub goal's weighted value is determined (WV $= 8 \cdot 0,3 = 2,4$). The sum of the two sub goal's weighted values multiplied by the main criterion's weight factor (WF $= 0,1$) leads to the intermediate values ($(7+2,4) \cdot 0,1 = 0,94$). In the table's last row the sum of the intermediate values, the benefit values are displayed per engine. The order of suitability of the three engines for the development of a game with the defined requirements is following:

Lowest Suitabilty	Benefit value: $8,94$	GDevelop
Medium Suitabilty	Benefit value: $9,1$	Unreal Engine
Highest Suitabilty	Benefit value: $9,8$	Unity

3.3.3 Final Game Engine

The comparison of the three game engines GDevelop, Unity and Unreal Engine leads to a clear outcome. The result from the benefit analysis 4 shows that Unity is the most suitable engine for this work and will therefore be used as platform for the game's implementation. Starting from 2017 Unity has released a new version every year with multiple Updates available [96]. Initially, it was intended to use the latest version to implement the game. In the test phase it was detected though that the newest version is incompatible with an essential plugin [86] that serves

[4] the easier to use, the higher the rating

to connect the Movesense sensor with the engine. This led to the transition to an older version. The game is now developed in the editor version 2019.4.40f1 [84]. In Unity's project manager empty C# scripts can be created and opened with Visual Studio 2015 or later that include C++ Tools component [84]. For this work the Microsoft Visual Studio 2022 version was used [66]. The engine works with a left-handed world space's coordinate-system, introduced in subsection 2.4.2.

3.4 Game Design

In order to achieve high children's motivation to exercise regularly, the game's design is of a great importance. Therapy can become boring when it consists of repetitive and intensive activities [27]. Demotivation and deconcentration may be the result, leading to the loss of the therapy session's effectiveness [27]. However repetitive and intensive training is important for motor learning. The options of game concepts are presented in subsection 3.4.1. To evaluate the concepts, a hospitation and an expert discussion were conducted, as described in subsection 3.4.2. Finally, a game concept was chosen that can be implemented in the selected game engine and meets the therapy requirements. It is presented in detail in subsection 3.4.3.

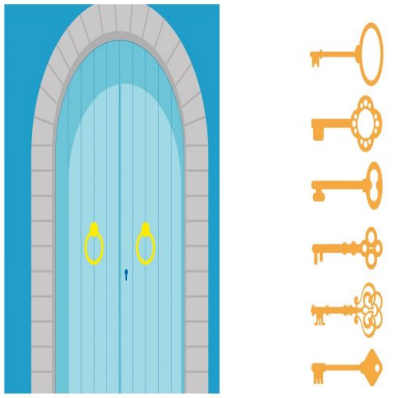
3.4.1 Game Concepts

The specific requirement in this context is to train supination and pronation movements of the hands. Therefore, situations requiring both motions have to be implemented. To furthermore ensure that children exercise on a regular basis, the game should be designed in an appealing and exciting way. From the previously reviewed design principles (2.3) the requirements considered to be the most important by the developer were derived. Furthermore, the decision was met to design the game in a 3D environment, due to the advantages stated in subsection 3.3.1. The requirements are divided into functional and non-functional requirements. They are recalled in table 5.

Table 5: Requirements for the SG's design

Functional requirements	
1	Difficulty levels
2	Precise instructions
3	3D graphics
Non-functional requirements	
4	Motivational factors like feedback and/ or reward
5	Interactivity

Before the elaboration of a game concept under consideration of the design principles and the overall system's requirements listed above in table 2, various approaches were investigated. Therefore everyday-life-situations using supination and pronation were collected to brainstorm different design ideas. The four most preferable ideas are shown in figure 8.



(a) Locking and unlocking doors[a]

[a]Designed by Katharina Lorenz in course of the SHArKi project in 2020

(b) Box of bricks [107]

(c) Feeding animals [70]

(d) Butterfly net [79]

Figure 8: Visualisation of the ideas for the game concepts

The first idea for the gaming concepts is locking and unlocking doors with a key 8a, differing in cognitive levels by variously shaped keys. The second idea is picking up bricks, rotating them and putting them into the holes of a construction kit 8b. The third option is feeding animals by pouring beverages or food into a food bowl 8c. The fourth option is rotating a butterfly net to catch items that are flying around 8d. To evaluate which game concept should be processed further, an expert interview was held during a hospitation in a physiotherapy practice.

3.4.2 Hospitation and Expert Discussion in Physiotherapy Practice

To get an insight into the movement patterns of children with CP, a work shadowing in the context of treatment and expert discussion was performed. Therefore, the physiotherapy practice "Praxis für neurologische Rehabilitation"[5] by Dagmar Siebold was visited. The two-year-old patient was observed during his exercise session with his therapist. An unimanual task was performed to get a first impression on how the child works with its affected hand. The child was given little objects in its hand and was supposed to put them into the box by supinating and opening the hand. For this exercise, the child needed the therapist's help to initialize the movement. Even though the child was categorized in a MACS-level II, it had difficulties with handling the exercises on its own and lost interest in the task quickly.

Subsequently, in an expert discussion with three therapists and the child's mother, the previously elaborated game concepts and the form of therapy were reviewed. According to the therapists, supination and pronation are the most challenging movements to be trained for the patients with CP. This can be explained by the natural holding of the affected hand that is, due to the spastics, usually in a supination tendency. The parent's child and the therapists agreed on the need to supervise the child whilst practising if the home therapy system were based on unimanual practice. It is necessary because constant support is required for initializing the movements. To avoid the constant need for supervision, the decision was met to design a bimanual therapy system, including a tangible device, held by the patient in both hands. This solution grants the child independence because the healthy hand can support the affected one. The movement on its own without the support of an external person is difficult and challenging for the patients. The inclusion of a tangible device is supposed to make the task easier and more realistic to perform. According to the therapists the age range $2-4$ is difficult to train, since the children quickly stop focusing and lose their motivation. Therefore, the first gaming concept with the key, locking and unlocking doors, seemed not exciting enough to catch the children's attention. The rotation of shapes and putting them in holes seemed too difficult, since it is hard for the children to hold still in certain angles. The third concept puts too little attention to the pronation movement. The most favoured concept was the fourth, rotating a butterfly net to catch items. It is age-appropriate, gives the option to use visual effects to grab the children's attention and to offer variety in the design. The concept was selected and is further detailed explained in the following subsection 3.4.3.

The therapists were additionally asked which information they were interested in storing if there was a monitoring file 2.3. Firstly they mentioned their interest in the times required for performing a training session in order to be able to

[5] `https://www.kinderphysiotherapie-praxis.de/`, last accessed: 24.11.2022

compare the time slots. By requiring less time for a training session in the long run, a facilitation in the movement could be concluded. Equally important is the data whether the movement itself was executed properly and its ROM. Besides that, the amount of executed movements that have been completed is of interest. Finally asked in further therapeutic needs, they pointed out the importance of choosing a diameter of the tangible device, suitable for the children's hands.

3.4.3 Final Game Concept

The elaboration of the final game concept was done with respect to the conditions from the main feature list (table 2) and the design requirements (table 5). The decision was made to develop the game as a bimanual training and to continue working with the butterfly net concept, introduced in subsection 3.4.1. Both choices were based on the opinion of the experts after the discussion described in subsection 3.4.2. In the following, the game concept and the game system's setup is described. The detailed development process with the more detailed concept's elaboration and the technical applicability is described in subsection 3.5.1.
The selected concept was adjusted due to the dependency on the availability of free assets in the Unity Asset Shop . Instead of using a butterfly net to catch items the adapted basic idea of the game is to make the items disappear by touching them with a stick. In reality the child holds a tangible device, a bamboo stick, as replacement for the stick on the screen. The bamboo stick is supposed to help the child to imagine the gaming situation, to have something to hold on to, and to let the healthy hand support the affected one in the movement process. To ensure a clean performance of the exercise, the mobile phone screen is set up on a table. The child sits in front of it on a chair with its elbows and forearms laying on the table. In the middle of the screen, the vertically aligned stick is shown in the starting position grounded on a surface. When the child pronates the right hand and supinates the left hand at the same time, the stick rotates to the left, at the same angle as the child hands' movement. When the upper palm of the left hand lays flat on the table, the maximum ROM is reached, the stick leans in an angle of 90° meaning it is horizontally aligned. The same applies to the supination of the right and pronation of the left hand. The stick rotates to the right reaching the maximum motion when the upper palm of the right hand lays flat on the table and the stick points horizontally to the right. The implementation of different motor game difficulty levels is required for the game's design, as first stressed in subsection 3.2. Three motor levels are therefore implemented, based on the Mini-MACS levels (see 2.1.1). The difficulty of the movement led to the choice restraining the game to children in the MACS levels I to III. For more severe courses like in patients classified in level IV or V, the movements supination and pronation are hard or impossible to perform. Even children in level three might require support throughout the training. With the

help of a parent or therapist the child's motion can be guided. The features of the levels differ in the ROM (ROM). The appearing items are positioned at differently sized angular spans away from the stick. These angles are set by the developer. They are adapted from literature in which the ROM of children with CP was measured during pronation and supination movements [4]. The angle is referred to as α and the three motor levels are summarised in table 6. In the game two playing levels correspond to each of the three covered MACS levels. The first game level is the basic level and the second one a cognitive level. A child classified in level III ends the game after level two. A child classified in level II has the chance to continue to play until it reaches level four whilst a child classified in level I can complete all six levels.

Table 6: Motor Levels

MACS Level	Game Levels	Mini-MACS characteristics	ROM
I	5&6	Handles objects easily and successfully	$-80° \leq \alpha \leq 80°$
II	3&4	Handles most objects, but with somewhat reduced quality and/or speed of achievement	$-40° \leq \alpha \leq 40°$
III	1&2	Handles objects with difficulty	$-24° \leq \alpha \leq 24°$

After the application is launched, a button can be activated to start the game in the first level. One item appears on the screen, with its position alternating right and left from the stick. It is aligned on the upper half of the circumference with its origin on the surface where the stick is placed. When the child rotates the stick and it collides with the item, the item disappears and a new one appears on the opposite side. The stick has to be rotated back into the starting position and further to destroy the new item on the other side. For every successfully removed item, the player earns a point. The score is counted and displayed on the screen. During the first phase of the game, the items appear close to the starting position, so that the supination and pronation motion performed is small. With each successful completion of a sequence the difficulty is increased by enlarging the angle. In each level eight items are to be destroyed. If this cannot be achieved, there is always the possibility to return to the main menu via a button on the mobile phone's screen and to exit the game from there. If all eight items were destroyed successfully, a "Feedback screen" automatically appears praising the player and granting the opportunity to replay the level or to continue with the next higher level. The feedback is intended to serve the non-functional requirement of motivational factors earlier presented in table 5. To be mentally demanding, the higher levels are implemented as cognitive levels. Instead of one item at a time, two items appear on the screen, one positioned

right and one positioned left from the stick. One of the items is yellow whereas the other one is blue. The player's task is to only hit the yellow ones. If the yellow item is hit, both items disappear and two new ones appear in the next positions. If the blue item is hit instead, a message is displayed giving the player the feedback that the wrong item was touched. After the appearance of a total of 16 items and the successful destruction of the 8 yellow ones, the cognitive level is over. Again the "Feedback screen" appears with the selection options given to either replay the cognitive level, to go back to the main menu, or to continue with the next higher difficulty level with bigger ROM. The same counts for game levels three, four, and five. At the end of level six, there are only two selection options to replay the level or to go back to the main menu to then quit the application.

3.5 Software Engineering

In this section it is described how the V-model, introduced in subsection 3.1, was applied to the software development of the system Movement Game. The framework for the process was examined in subsection 3.3 and 3.4, with the determination of the SGE and the game concept. The final goal of the software development process is to proceed the game concept into an actual application. Therefore, the rough draft is specified into the fine draft, that is then implemented and tested. As stated in subsection 3.1, the complex main system is decomposed into subsystems. Initially, the game concept is implemented on the computer with a placeholder for the sensors. Following, the component Android cell phone is considered. Then, the Movesense sensor's software is being processed and finally included into the overall system as motion input.
The first subsystem deals with the general game flow and its details, described in the following subsection 3.5.1. The second system Android environment with the components building of an Android application and enabling developer mode on an Android cell phone is introduced in 3.5.2. It is required to later test the third system's sensor software components that are described in subsection 3.5.3.
For each of the mentioned systems, the tasks **requirement elicitation**, **system architecture**, **implementation of system elements**, and **system integration and verification** are performed in the corresponding subsections. The highest core task **validation and transition** is only partly considered. This work is restricted to the verification if the system was built correctly and if all requirements from the main-feature list 3.2 were met. The transition and verification of the customer's satisfaction is due to limitations not examined.

The technical drafts, created in the core task **system architecture** are visualized as flowcharts. In these flowcharts, differently shaped blocks are connected. Each shape represents one type of block. The rectangles with rounded edges represent the start and end of the process. The rectangles with sharp edges indic-

ate processes. The parallelograms represent inputs and outputs. The rhombuses stand for conditions or decisions that are answered with a yes or no [68].

3.5.1 Component Game Flow

To begin with, general requirements are defined in the V-model's first core task, the **requirements elicitation**. The actual application is still ignored and only the black box view is used to describe how the system should implement the previously determined requirements. The requirements for the game's design, listed in table 5 and the selected game concept, described in subsection 3.4.3 are now further processed. The subsystem game flow's requirements are established in table 7.

Table 7: Requirements for the Game

Obligatory requirements	
1	Stick that can be set in motion
2	Items that appear
3	Items that disappear when triggered
4	Game running on Android operating system
Optional requirements	
5	Effect when item is destroyed
6	Point system
7	Display Messages

The obligatory requirements for the game flow are a stick that can rotate to the right and left, objects that appear and disappear when touched by the stick, and it needs to be runnable on an Android operating system. Optional requirements are the occurrence of effects when the object is destroyed, a point system, and the display of information text. These wishes were determined on the basis of the non-functional requirements for the game's design (5).
After the specication of the game flow's requirements, the process continues with the V-model's core task **system architecture**. The general game flow of the Movement Game is following described and the linking of the different scenes is presented to simplify understanding the subsystem. The game flow's technical draft is visualized in flowchart 9 by the application's scene management. At this stage, the subsystem game flow is only configured for the computer. The scene management is controlled by mouse tabs on the computer screen's buttons. However it is already mentioned that the process on the Android application is almost identical. It only differs in the control of the scene management via finger input on the cell phone's touch screen.

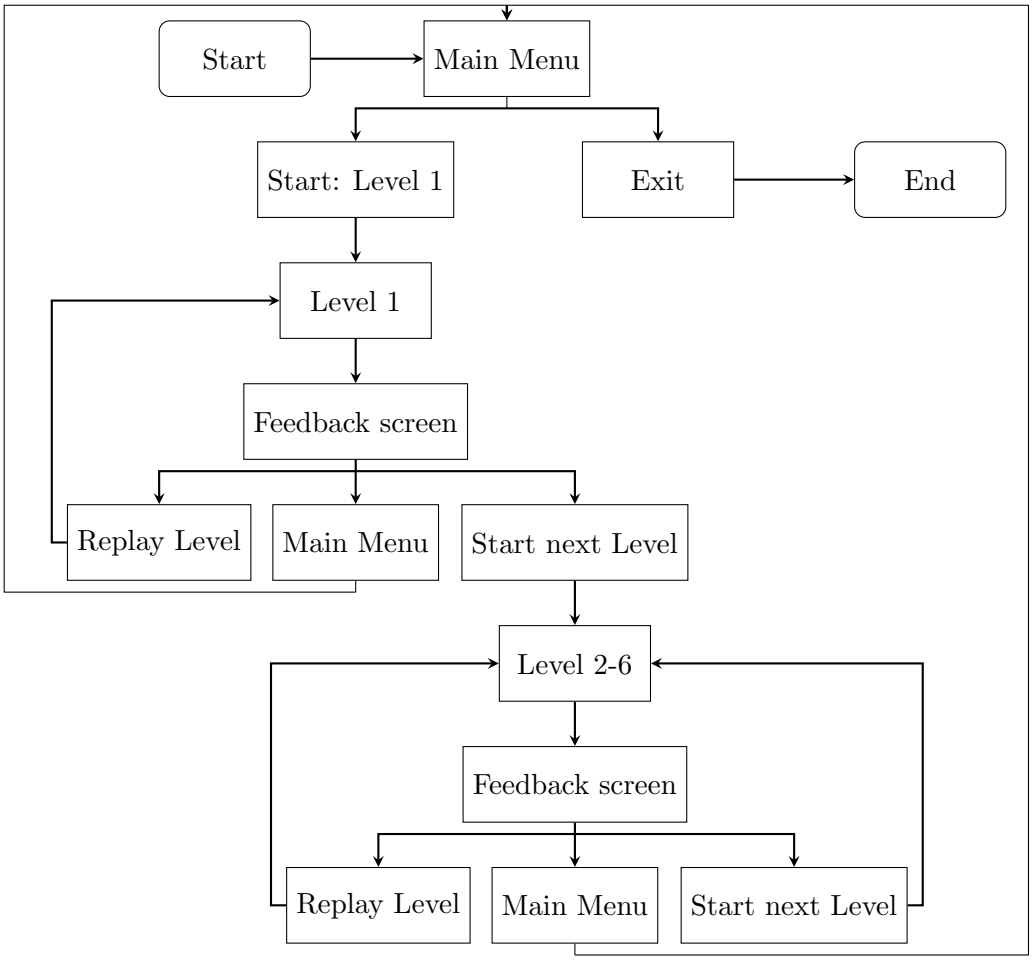


Figure 9: Component Game Flow

The subsystem's architecture and the application's usability is further explained. When the application is opened the "Main Menu" screen appears, explaining how the game works and giving the option to select a button: "Start: Level 1", or "Exit". Pressing one of the buttons, the **MainMenu** script calls the Unity class **SceneManager**. Via the "Exit" button the application is closed. The "Start: Level 1" button leads to the display of the basic level for children categorized in MACS level III. The component game flow with its subsystems is described below in flowchart 10. After the completion of the first level the "Feedback screen" appears. From here, the level can either be replayed by activating the "Replay Level" button, the "Main Menu", or the next level can be opened. This is applicable to all six implemented levels, with the exception that after the sixth level the "Start next Level" button is not displayed. When the button "Start next Level" is chosen after the first level, the cognitive level for MACS level III is executed. How it differs from the basic level is explained later on, after the introduction of the component game flow with subsystems. The "Feedback screen" of the second level grants the same opportunities as the first "Feedback screen" with the exception that the next executed level (level three) is the first

level designed for children with MACS level II. The following level four is the cognitive level for children categorized in MACS level II. The game's fifth level is the first level for children in MACS group I, and the game's sixth level is the cognitive level implemented for them.

Following, the game flow's subsystems are explained. The active game experience involving movement starts when executing the first level. To simplify the complex overall system, the motion input by the sensors was initially neglected. Instead, a placeholder was used as motion input. In the test phase, the right and left arrow keys were used to control the stick's movement. The stick is positioned on a flat surface and rotates to the left and to the right when the movement is activated. Flowchart 10 specifies the rough process of the game flow with its subsystems in the test phase. It describes the **sytem architecture** of the subsystem game flow in detail.
When the first level is started, in the **PrefabManager** a for-loop is executed with the counter $i = 0$. For each loop, the decision is checked, if the counter is still smaller than eight. If so, an item is spawned at the counter's position by the method **Spawning()**. These positions and the item are stored in a list. As soon as the player submits an input by using the left or right arrow key, the **RotatePaddle** script's method **RotateWithKeys()** is addressed. The stick is set in motion. Once it starts rotating it is checked if the rotation aims towards the instantiated item. If the rotation is executed in the wrong direction the next decision is checked if the stick is touching the surface. The surface and the stick have an activated Trigger that allows to use **OnTriggerEnter(Collider)** method [89]. The collision of the two objects calls the script **ScoreManager**'s method **WrongMovement()** that displays a message on the screen informing the user to rotate the stick into the opposite direction. The spawned items also have an activated trigger and the script **DestroyItems** attached to them. Its Triggermethod is activated once a collision with the stick takes place leading to the multiple co-processes. Once the object is destroyed, confetti comes out of it. The **ScoreManager**'s method **IncreaseScore** adds a point to the account and the **Spawning()** method is recalled, adding one point to the counter. After the process passes the eighth time, the boolean **checkLevelDone()** returns a true, leading the Spawning() method to call the GameOverScreen(). The total amount of points is then displayed and feedback is given. The process continues from here, with the possibility to select between replaying the level, playing the next level, and exiting the game as described in subsection 3.5.1. The third and fifth level go through the same process, differing slightly in the **PrefabManager**. The positions stored in the attached lists have other angular ranges, as stated earlier in table 6.

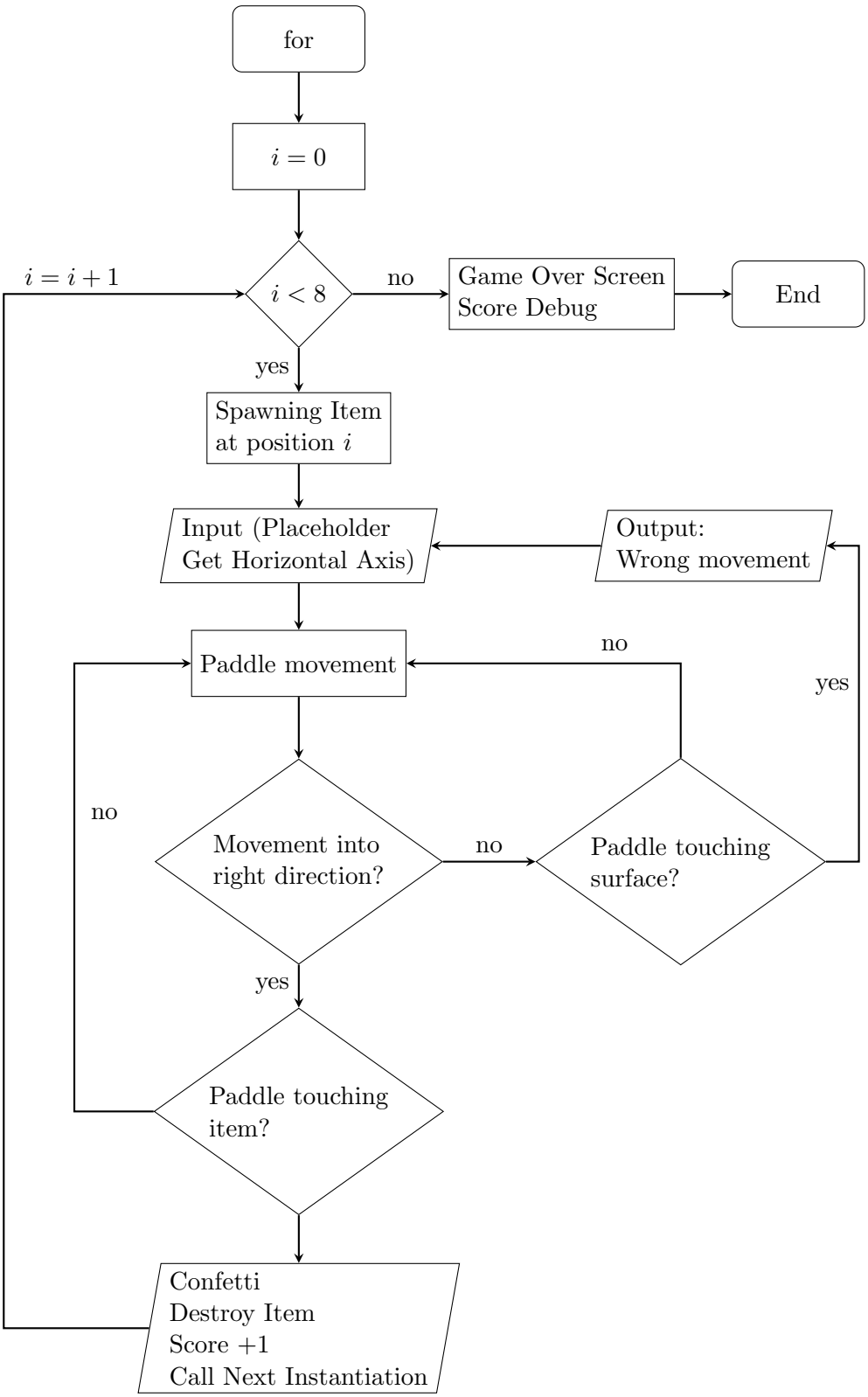


Figure 10: Component Game flow with subsystems

The second, fourth, and sixth level go through the same processes as the basic levels with additional adjustments. The cognitive levels' processes differ in the amount of lists that are utilized from the **PrefabManager**. Instead of one list

that stores the positions and an item, two lists are used. **Spawning()** then instantiates two objects at two positions. The following process remains the same except for the fact that the blue prefab has the script **WrongItems** attached to it with an activated trigger. When the stick collides with the item, the information that the wrong item was touched is displayed.
The achievement of the desired motion input by the Movesense sensors is described further below in subsection 3.5.3. The game flow remains the same, but the Movesense sensor as input is interchanged with the process "Placeholder Get Horizontal Axis".

The technical drafts are subsequently used with the main-feature list by the developer to implement the game in the core task **implementation of system elements**. The developed scripts and methods for the game flow, including the scene management and the game flow in details, are summarized in table 8. The methods **UpdatePaddleRotation()**, **InitialText()** and **ConnectionSuccessfull()** are relevant for the motion control by the sensors and will be referred to later-on. In appendix B there is a link through which the scripts can be accessed [6].

Each component of the subsystem game flow is regularly tested and validated in the process to ensure that the specified requirements were met. The individual requirements for the game were implemented one after the other. To do so, code was written and debugged. In Unity engine the code is written in .cs files that can be opened in the game engine via Visual Studios. When error messages appeared, solutions were searched for to fix the errors and to eventually eliminate them. Once the code was error-free, the component was tested in Unity's game environment. For this, the code was attached to the engine and the game was played. Playing the game led to three possible situations. The first possible situation is the appearance of console notifications or errors in Unity. If this happens, it is necessary to debug and to modify the code in order to retest it. The second situation is the failure to fulfil a set requirement. It is possible that the code is error-free but does not meet the target of the obligatory requirements. Then again the code needs to be modified and retested. The third situation is the desired outcome. No error messages are displayed and the component serves its specified purpose.

[6] \AbgabeBachelorarbeit\Applikation\PaddleGame_BachelorThesis_MirbachRebekka \Assets\Scripts

Table 8: Game's scripts and methods

Script	Methods	Comment
DestroyItems	OnTriggerEnter	When item triggered: Destroy Gameobject, Display Effect, Start Coroutine, Call PrefabManager, Call ScoreManager
GameOver	Setup	Display level and score
	RestartButton, NextLevelButton, MainMenuButton	Call SceneManager>LoadScene
MainMenu	Start	Display connectedText
	ExitButton	Disconnect sensor, Quit Application
	StartFirstLevel	Call SceneManager > LoadScene
	InitialText	Display connectedText when sensor is not connected
	ConnectionSuccessfull	Display connectedText when sensor is connected
PrefabManager	init	Defining lists and cases for levels
	Spawning	Instantiate Object at position of list
	checkLevelDone	boolean
	GameOverScreen	Call GameOver > Setup
RotatePaddle	Update	RotateWithKeys
	RotateWithKeys	Movement horizontalInput Rotate/ ScoreManager > WrongMovement
	OnCollisionEnter	Call ScoreManager > WrongColor
	UpdatePaddleRotation	Rotate Paddle by quaternion
ScoreDisplay		Storing scoreText and levelText
ScoreManager	IncreaseScore	Score+1, UpdateScoreDisplay
	UpdateScoreDisplay	Change scoreText and levelText
	WrongMovement	Change scoreText
	WrongColor	Change scoreText
WrongItems	OnTriggerEnter	Call ScoreManager > WrongColor
	OnTriggerExit	Call ScoreManager > UpdateScoreDisplay

The above described methods and functions were tested as demanded in the V-model's core task **system integration and verification**. Test cases were established and performed to test the individual functions' usability. The test cases

check whether the requirements defined at the beginning of the project have been met by the technical draft's implementation. As an example, the first test case for the scene management on the computer is specified in table 9. It serves to check if the scene management works accurately when the computer mouse is used as input. The test case is called "Scene management computer" (SMC). Later, the test case for the Android application's scene management with finger input will be referred to as "Scene management Android" (SMA) 16.

Table 9: Test Case Example Scene Management Computer

Test Case ID: SMC-1 **Priority: 1**	**Purpose:** to test the mouse input's function
Preconditions:	The game flow is played on computer
Inputs:	Press button by mouse click
Expected results:	Scene is opened

For each of the test cases concerning the computer's scene management the results are reviewed in table 10.

Table 10: Results Scene Management Computer Test Cases

Scene Management Computer		
Test Case ID	**What was tested?**	**Passed** (✓) **Not passed** (x)
SMC-1	The mouse input's function when game flow played on computer	✓
SMC-2	Corresponding scenes opening trough mouse input	✓
SMC-3	Game quitting when "Exit" button is pressed through mouse input	✓

All test cases were passed and the scene management was therefore successfully implemented.
Following the game flow in detail was tested. In table 11, the example of the first test case for the test series "Game Flow Computer" (GFC) is given. Its purpose is to check if the stick on the screen rotates to the left when the left arrow key is used as motion input. Further unit test cases were executed. The subsystem's requirements and all game flow components are profoundly tested. The test completion report with the test's purposes and results is summarized in table 12.

Table 11: Test Case Example Game Flow Computer

Test Case ID: GFC-1 **Priority: 1**	**Purpose:** to test the stick's movement to the left
Preconditions:	Stick implemented and game played on computer
Inputs:	Press left key arrow on computer
Expected results:	Stick is rotated to the left

Table 12: Results Game Flow Computer Test Cases

Basic Game Flow Computer		
Test Case ID	**What was tested?**	**Passed** (✓) **Not passed** (x)
GFC-1	The stick rotates to the left when input on computer left arrow key	✓
GFC-2	The stick rotates to the right when input on computer right arrow key	✓
GFC-3	The stick's movement to the left is limited to 90°	✓
GFC-4	The stick's movement to the right is limited to 90°	✓
GFC-5	WrongMovement text displayed when surface is touched	✓
GFC-6	Item spawning	✓
GFC-7	Item appears at desired position	✓
GFC-8	Item triggered when touched by stick	✓
GFC-9	Item disappears when triggered by stick	✓
GFC-10	Confetti spawning at item's position	✓
GFC-11	Score +1 displayed on screen	✓
GFC-12	New item spawning at following position from position's list	✓
GFC-13	Feedback screen appears after destruction of 8th item	✓
GFC-14	Feedback screen displays motivational text and buttons to select other scene via scene management	✓

The additional testing for the cognitive level is summarized in table 13.

Table 13: Results Game Flow Cognitive Level Computer Test Cases

Cognitive Level Computer		
Test Case ID	**What was tested?**	**Passed** (✓) **Not passed** (x)
GFC-15	Yellow item spawning	✓
GFC-16	Yellow item appears at first position	✓
GFC-17	Blue item spawning	✓
GFC-18	Blue item appears at first position	✓
GFC-19	Yellow and blue item triggered when touched by stick	✓
GFC-20	Yellow item disappears when triggered by stick	✓
GFC-21	Confetti spawning at yellow item's position	✓
GFC-22	Score +1 displayed on screen	✓
GFC-23	Message WrongItems displayed when blue item triggered	✓
GFC-24	Blue item disappears when yellow item is triggered by stick	✓
GFC-25	New yellow and blue item spawning at following positions from position's lists	✓
GFC-26	Feedback screen appears after destruction of 8th item	✓
GFC-27	Feedback screen displays motivational text and buttons to select other scene via scene management	✓

The test procedures were all executed and passed testing. Six of the seven requirements derived in the subsystem's requirement elicitation (7) were satisfied. The requirement, if the game is running on an Android operating system, requires further proceedings and is therefore addressed in the following subsection 3.5.2.

3.5.2 Component Android Environment

After successfully implementing the game flow on the computer as described in subsection 3.5.1, the subsystem Android environment is considered. The requirements to be fulfilled are established first. They are listed in table 14.

Table 14: Requirements for the subsystem Android environment

Obligatory requirements	
1	Capability to run the developed game
2	Compatibility with the Movesense sensors
3	Bluetooth module to scan for the BLE sensor

The Android device that is used for the testing, meets the third requirement of having an integrated bluetooth module. The compatibility with the Movesense sensors is examined in subsection 3.5.3. In the following, the capability to run the developed game is investigated. Building an Android application in Unity is an iterative process. Flowchart 11 illustrates the individual steps of the workflow.

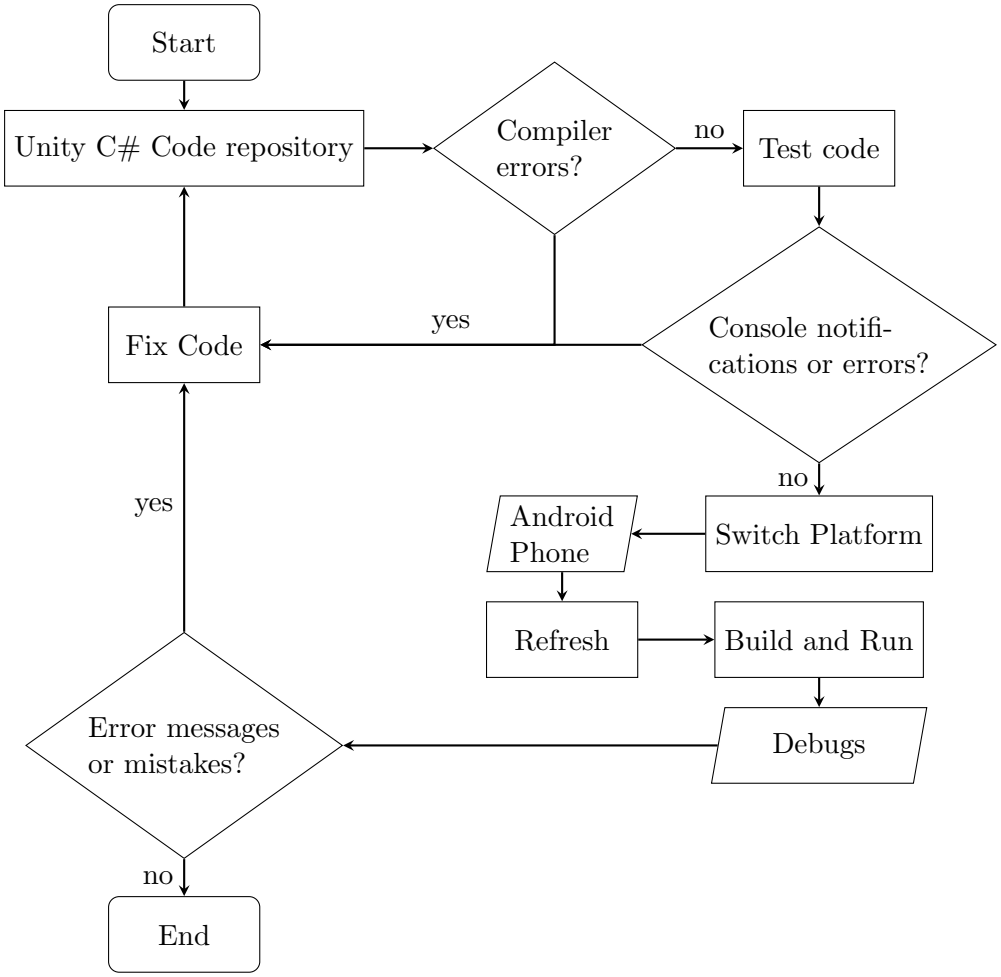


Figure 11: Building an Android Application

It is aimed to run an application developed in Unity on an Android device. Therefore, firstly the game engine's environment needs further investigation. The process starts with checking the code repository for compiler errors. These are displayed in Visual Studios and prevent the code from being executed in Unity. If

there are errors the code needs to be fixed. Once they are fixed the code can be tested in Unity. This is realized by attaching the code to Unity and running the game. Unity's console then displays notifications or errors if the code is not compatible with the game or cannot be run. In this case the code has to be fixed. If not the process continues.
Following the platform on which the application is supposed to run needs to be selected. For this work the platform was switched to Android. An Android cell phone with activated developer options is required to serve as input and output. Flowchart 12 visualizes how the phone is set into development mode as a side process: To enable developer mode on the Android phone the settings system has to be opened. Under the tab "About phone" the build number can be found. By touching the build number seven times, the developer mode is enabled and the message "You are now a developer!" is displayed on the screen. The activation causes the button "Developer options" to appear in the system settings. This option has to be enabled. Additionally, the USB Debugging needs to be activated and the USB configuration has to be set to MIDI. After completing this process, the phone can be connected to the computer via a cable to serve as an In- and Output. Once the phone is connected to the computer and the connected devices are refreshed, the application can be build and run. An .apk file is created that is automatically opened on the phone once the building is complete.

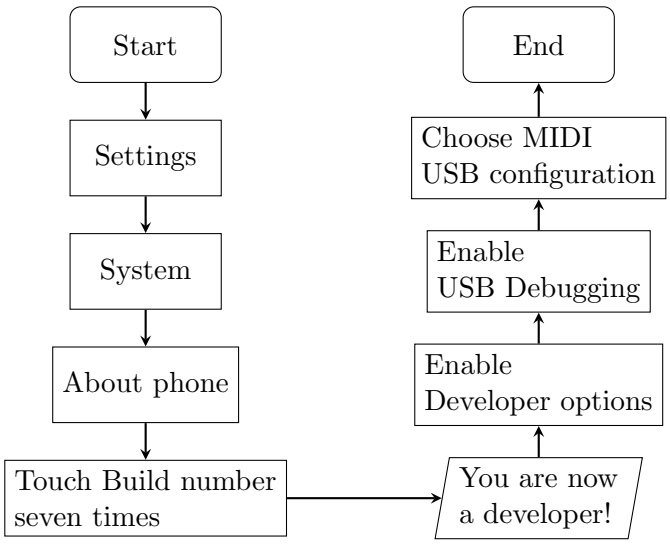


Figure 12: Enable Developer Mode on an Android phone

Via Android Logcat [65] the application's error messages and debugs from the scripts are displayed in Unity. By running the above described process several times, errors can be detected and corrected. Once all errors that occurred have been eliminated, the process is complete. The application works and can be used. The game implementation, elaborated and tested in subsection 3.5.1, is now used to build a test application. In the test application, the scene management's func-

tion can be verified. The test cases executed for the game flow's scene management on the computer 10 are adjusted and executed on the Android device. The exemplary test case is given in table 15.

Table 15: Test Case Example Scene Management Android Application

Test Case ID: SMA-1 **Priority: 1**	**Purpose:** to test the finger input's function
Preconditions:	The game flow is played on Android device
Inputs:	Press button with finger on touch screen
Expected results:	Scene is opened

All test cases and results concerning the game flow's scene management on the Android device are reviewed in table 16.

Table 16: Results Scene Management Android Application Test Cases

Scene Management Android Application		
Test Case ID	**What was tested?**	**Passed** (✓) **Not passed** (x)
SMA-1	Developed application launching on Android device	✓
SMA-2	Main Menu is displayed on Android device	✓
SMA-3	Dimensions of the buttons and the text well adjusted to Android device's size	✓
SMA-4	The finger input's function when game flow played on Android device	✓
SMA-5	Corresponding scenes opening trough finger input	✓
SMA-6	Game quitting when "Exit" button is pressed through finger input	✓

The tests were passed, and the subsystem's requirement to run the developed game 14 was validated and verified. However, the testing only concerned the scene management and not the game flow with motion input. This is tested after the Movesense sensors are successfully implemented into the overall system. The process visualized above in flowchart 11 was passed multiple times to connect the sensors to the engine and to ensure smooth motion transmission. The proper path to realize the motion input by the Movesense sensors is further described in the following subsection 3.5.3.

3.5.3 Component Sensor Software

The Movesense sensor's software is the game system's third subsystem. The motion control of the game is to be achieved by detecting and converting motion data from the Movesense sensor. The subsystem's requirements are specified in table 17.

Table 17: Requirements for the subsystem sensor software

Obligatory requirements	
1	Embedded Bluetooth module to be detectable for the Android device
2	Connection establishment with Android device
3	Data transmission via BLE
4	Disconnection from Android device

As mentioned in subsection 2.4.1, the Movesense sensor is equipped with Bluetooth Low Energy radio. The first requirement is thus satisfied. For the connection, subscription, and disconnection of the device from the Android application, a connector module between the sensors and the SGE Unity is required to be used. Movesense developed a plugin for Unity in 2019 that is free to download in Unity's asset store [86]. A detailed description on how the plugin is implemented is given in appendix C. The plugin contains scripts with methods scanning for the sensors[7], connecting them to the engine[8] and casting for a subscription path. The subscription paths include linear acceleration, angular velocity, magnetic field, temperature, and more[9]. A description of how the various methods and events in the scripts work is provided by the company Kaasa Solution in the plugin's documentation[10]. The descriptions were followed in order to be able to scan for the sensors, to connect them with the Android application, to subscribe their motion data, and to disconnect them when the application is quit. When subscribing the motion data, the sample rate for the subscription path needs to be chosen. From the available sample rates, that are all listed in the documentation, the slowest one with 13 Hertz was chosen. It was tested in test cases, later listed in table 21, in which it has been found to be sufficient for the desired motion sequences.
Subsequently, the Movesense-Plugin contains example scenes, that were used to implement a simple application to test the connection establishment. The test cases are referred to with the ID "Movesense-Plugin" (MP). The first test case concerns the scanning process and is displayed in table 18.

[7] [86]: ScanController.cs
[8] [86]: MovesenseController.cs
[9] [86]: MovesenseController.cs public class SubscriptionPath
[10] Movesense-Plugin for Unity. Version 4. Description and usage for all methods and events. Path: follow #3 in the table Movesense-Plugin Unity in appendix C

Table 18: Test Case Movesense-Plugin Example Scenes

Test Case ID: MP-1 **Priority: 1**	**Purpose:** to test the Movesense-Plugin on scanning
Preconditions:	The Movesense-Plugin is implemented in the Android application and the application is played on Android device
Inputs:	Press button with finger on touch screen to start scanning
Expected results:	Movesense sensors within BLE reach are displayed

The other relevant test cases for the subsystem as well as their results are summarized in table 19.

Table 19: Results Movesense-Plugin Test Cases

Movesense-Plugin		
Test Case ID	**What was tested?**	**Passed** (✓) **Not passed** (x)
MP-1	Scanning process	✓
MP-2	Connection establishment	✓
MP-3	Disconnection	✓
MP-4	Subscribing linear acceleration data	✓

On basis of the successful connection establishment and the subscription of the linear acceleration data, the focus from here is placed on the game flow's motion control. The data of interest to later realize the movement is the linear acceleration data. It is converted into a quaternion to rotate the stick in the Android application. Using the accelerometer data instead of the angular velocity, derived from the gyroscope, prevents drifting. It also prevents the need to use a filter, as explained in subsection 2.4.1. For this application, the Movesense sensor's x-axis, visualized in figure 5a rotates around one axis only, the world space's z-axis 5b. Test trials have shown that the linear acceleration data serves the desired purpose for this movement.

To represent rotations, the choice was met to work with unit quaternions, allowing for efficient computation [94]. Furthermore, the calculations are simplified since Unity has its own Quaternion class in which the three dimensional orientation of an object can be stored and the relative rotation between two orientations can be described [90]. In the script **MovesenseSensors** the necessary operations are

executed to convert the sensor's linear acceleration data into a quaternion. The operations are summarized in flowchart 13.

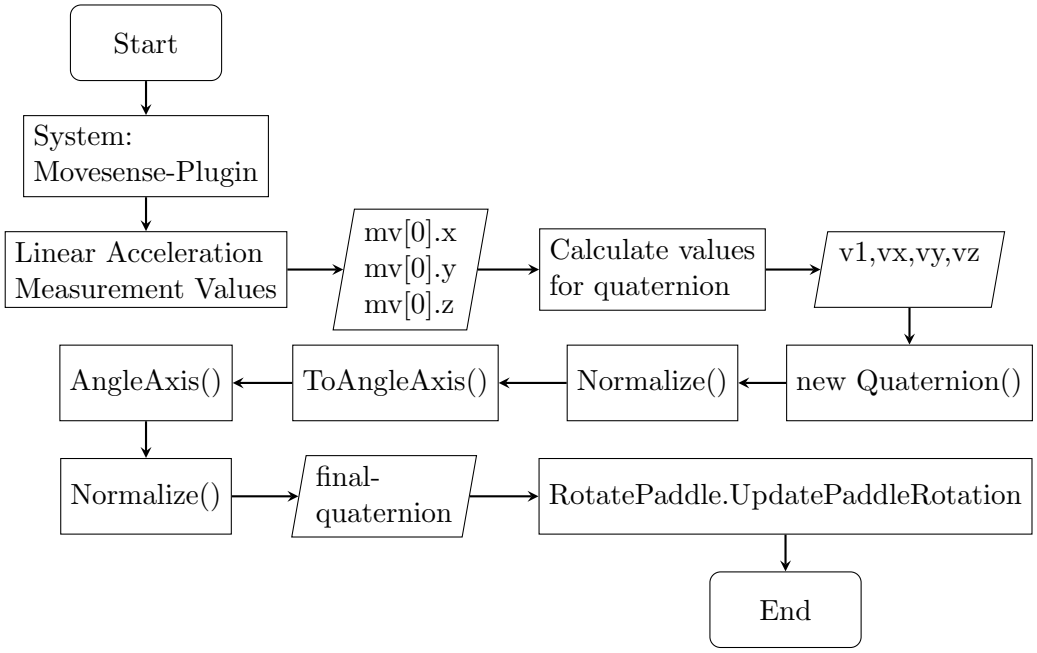


Figure 13: Movesense sensors as motion input

First, the BLE-IMU is scanned for and once it is found, connected to the application. The process is summarized in the flowchart under the term system Movesense-Plugin. The scanning process is started automatically when the application is launched. As long as the sensor is not connected yet or still connecting, the **MainMenu**'s method **InitialText()** is run and the following information is displayed on the "Main Menu" screen: "Please wait for the sensor to connect! The connection message will be displayed here". Once the sensor is connected, the **MovesenseSensors** script calls the **MainMenu**'s method **DisplayText()**. The text on the screen is thereby changed to "You can start playing now, your sensor successfully connected!". Both methods are listed in the implementation table 8. The decision was met to start the scanning automatically instead of using a scanning button like in the example scene. It makes the experience user-friendlier and reduces the amount of time that has to be waited before the game can be started. When the sensor is connected, the linear acceleration values are subscribed. This is realized in the **MovesenseSensors** script. The measurement values are composed of the entries double x, double y, and double z. They are updated by notification callbacks and the values are stored in the list **mv**[]. In Unity's quaternion method the four components x, y, z, and w are used to encode the direction and angle of rotation about unit axes in three dimensions [94]. The alignment of the sensor is thereby essential for the direction of motion.
As described in chapter 2.4.2, quaternions are composed of a scalar part s and a vector part v (1). The vector part is defined by the measurement values. The

earlier declared formula 2 is used to calculate the input parameters (further declared as x=v1, y=vx, z=vy, and w=vz) for the quaternion q, by applying the angle $\theta = 90°$, and the linear acceleration data n_x = mv[0].x, n_y = mv[0].y, and n_z = mv[0].z:

$$q = \underbrace{\cos\left(\frac{\theta}{2}\right)}_{v1} + i \cdot \underbrace{n_x \sin\left(\frac{\theta}{2}\right)}_{vx} + j \cdot \underbrace{n_y \sin\left(\frac{\theta}{2}\right)}_{vy} + k \cdot \underbrace{n_z \sin\left(\frac{\theta}{2}\right)}_{vz}. \tag{4}$$

The four values (v1,vx,vy,vz) are passed to Unity's quaternion method that constructs the quaternion **quaternion**. This quaternion is then normalized, meaning that it is scaled to a length of one [63]. Therefore one of unity's build-in methods is used. The **Quaternion.Normalize()** operation converts a quaternion that is passed into a quaternion with the same orientation but a magnitude of one [92]. In the next step, the variable is passed to the operation **Quaternion.ToAngleAxis()** that extracts an angle in degrees from the axis rotation [93]. The input is an angle as a float value and an axis as three-dimensional Vector [93]. **Quaternion.AngleAxis()** takes the angle and axis and creates a new rotation [91]. In test cases, it was determined that the bamboo stick rotates by twice the angle of the stick in the application. Therefore the passed angle parameter is divided by two to achieve the corresponding rotation of the bamboo stick in reality and of the stick on the screen. Afterwards the created quaternion is again normalized. This leads to the **finalquaternion** that is used to rotate the stick. With **Transform.rotation()** a GameObject is rotated by the stored quaternion in world space [95]. It allows to apply arbitrary rotations to every orientation and thereby prevents gimbal lock to occur. The operation is used in the **UpdatePaddleRotation()** method in the **RotatePaddle** script. The **RotatePaddle** and the **MovesenseSensors** scripts are attached to the the game object stick and the motion control's implementation by the sensors is thereby completed. When the application is quit by pressing the "Exit" button the sensor is automatically disconnected by the **ExitButton()** method 8. The test cases are referred to in the overall system's requirement verification in the following subsection 3.6.

3.6 Overall System's Requirement Verification

Lastly, the entire Movement Game system is technically verified. It is checked and summarized whether the subsystems' requirements, listed in table 7, 14, and 17 and the overall system's requirements, listed in the main-feature list 2 are all fulfilled. In the game flow's requirements test cases, six of seven requirements were verified when playing the game on the computer 12. Hereby the requirement that the game is running on an Android operating system could not yet be verified. This requirement was partly verified in the Android subsystem's test phase in

subsection 3.5.2. The scene management's functioning on the Android device was validated, as stated in table 16. The detailed game flow on the Android application needs the implementation of the three subsytems into the overall system and is therefore lastly conducted. Following, the requirements derived from the Android environment and the sensor software are checked in test cases on the Android device. The cases concerning the sensor software in the final application are labelled with the abbreviation MPA. The first test case is the automatic scanning for the Movesense sensor, described in table 20.

Table 20: Test Case Movesense-Plugin Final Application

Test Case ID: MPA-1 **Priority: 1**	**Purpose:** to test the automatic scanning in the Movement Game application
Preconditions:	The Movesense-Plugin is implemented in the Movement Game and the application is played on Android device
Inputs:	Application launched and scanning process started automatically
Expected results:	Movesense sensor within BLE reach connects

The results of the test cases are summarized in table 21.

Table 21: Results Movesense-Plugin Final Application Test Cases

Movesense-Plugin		
Test Case ID	**What was tested?**	**Passed** (✓) **Not passed** (x)
MPA-1	Scanning process	✓
MPA-2	Connection establishment	✓
MPA-3	Display InitalText() when sensor still disconnected or connecting	✓
MPA-4	Display ConnectionSuccessfull text when sensor connected	✓
MPA-5	Subscribing linear acceleration data to be converted and to serve as motion input	✓
MPA-6	Sample rate 13 Hertz sufficient for movement without jitters	✓
MPA-7	Disconnection when "Exit" Button activated	✓

All requirements from the sensor software's requirement elicitation 17 were hereby verified. The connection establishment with the Android device was successfully implemented, the desired data to serve as motion input is transmitted via BLE and the sensor is disconnected from the Android device when the application is quit. Now eventually it is considered if the detailed game flow runs in the Android environment with the Movesense sensor as motion input. One test case is given as example in table 22. The test completion report is summarized in table 23.

Table 22: Test Case Example Game Flow Android application

Test Case ID: GFA-1 **Priority: 1**	**Purpose:** to test the stick's movement
Preconditions:	Game flow implemented in Android application and played on Android device
Inputs:	Motion input by the Movesense sensor
Expected results:	Stick rotates into direction according to motion input

Table 23: Results Game Flow Android Application Test Cases

Basic Game Flow Android Application		
Test Case ID	**What was tested?**	**Passed** (✓) **Not passed** (x)
GFA-1	The stick rotates to the left when the sensor is rotated to the left	✓
GFA-2	The stick rotates to the right when sensor is rotated to the right	✓
GFA-3	The stick's movement to the left is limited to 90°	✓
GFA-4	The stick's movement to the right is limited to 90°	✓
GFA-5	WrongMovement text displayed when surface is touched	✓
GFA-6	Item spawning	✓
GFA-7	Item appears at desired position	✓
GFA-8	Item triggered when touched by stick	✓
GFA-9	Item disappears when triggered by stick	✓
GFA-10	Confetti spawning at item's position	✓
GFA-11	Score +1 displayed on screen	✓
GFA-12	New item spawning at following position from position's list	✓
GFA-13	Feedback screen appears after destruction of 8th item, displays motivational text and buttons to select other scene via SMA	✓

All executed tests passed. Subsequently, the test cases for the cognitive levels were carried out, as shown in table 24.

Table 24: Results Game Flow Cognitive Level Android Application Test Cases

Cognitive Level Android Application		
Test Case ID	**What was tested?**	**Passed** (✓) **Not passed** (x)
GFA-14	Yellow item spawning	✓
GFA-15	Yellow item appears at first position	✓
GFA-16	Blue item spawning	✓
GFA-17	Blue item appears at first position	✓
GFA-18	Yellow and blue item triggered when touched by stick	✓
GFA-19	Yellow item disappears when triggered by stick	✓
GFA-20	Confetti spawning at yellow item's position	✓
GFA-21	Score +1 displayed on screen	✓
GFA-22	Message WrongItems displayed when blue item triggered	✓
GFA-23	Blue item disappears when yellow item is triggered by stick	✓
GFA-24	New yellow and blue item spawning at following positions from position's lists	✓
GFA-25	Feedback screen appears after destruction of 8th item	✓
GFA-26	Feedback screen displays motivational text and buttons to select other scene via scene management	✓

The requirements derived for the subsystem game flow have hereby been validated and verified. After decomposing the overall system into smaller work packages all system elements were individually tested and verified. Then all the elements were reintegrated into the Movement Game. With that the implementation and integration of the system was completed. The reviewed test cases serve all established requirements from the subsystem's requirements elicitations 7, 14, 17 and the overall system's main-feature list 2. The system's validation and V-model's highest core task is therefore complete.

Lastly, the overall system was tested by five adults, as well as a four-year-old child. The six individuals have no motor impairments. The test cases are not to be understood as user-tests but serve to check the technical functioning of the system. They are reviewed in table 25 and are labelled OS (Overall System).

Table 25: Results Overall System's Test Cases

Basic Game Flow Android Application		
Test Case ID	**What was tested?**	**Passed** (✓) **Not passed** (x)
OS-1	Easy to understand instructions	✓
OS-2	Scene Management functioning	✓
OS-3	Smooth and bias-free motion of the stick on the screen when the bamboo stick with the sensor attached to it is supinated or pronated	✓
OS-4	All game flow application test cases (GFA $1-26$) functioning	✓
OS-5	All Movesense-Plugin final application test cases (MPA $1-7$) functioning	✓

By passing all tests, the overall system Movement Game was technically validated and verified. The results are commented in the following section 4 and then critically discussed in section 5.

4 Results

In this chapter, the results of the developed application are presented in visual form. In addition, reference is made to the usability, which was already explained in subsection 3.5.1. The Movement Game application is saved as an .apk file[11] and is transferred to an Android device via bluetooth or a cable connection. Once the application is launched, the "Main Menu" screen appears. It is visualized in figure 14. The explanation text displayed on it refers to spheres, because they are the elected items to be spawned for the game.

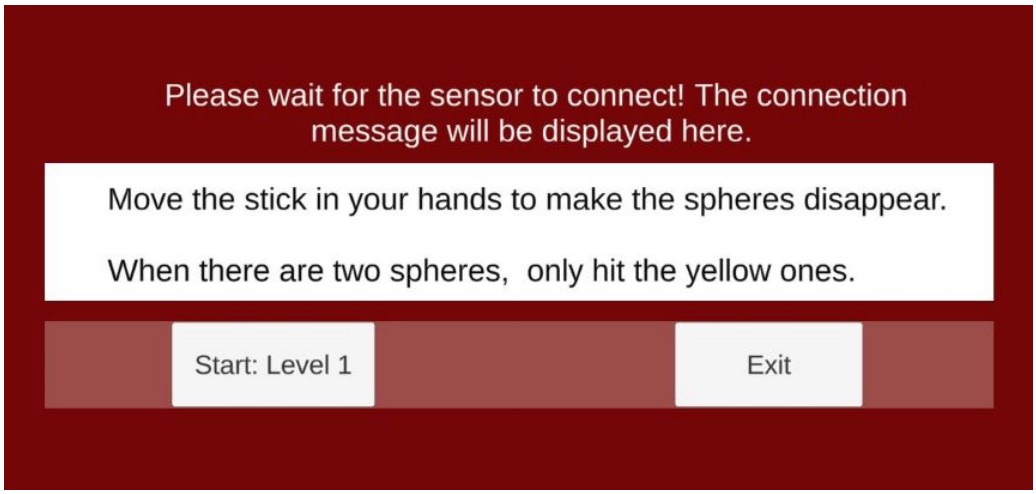


Figure 14: "Main Menu" screen when sensor disconnected

When the battery is placed into the Movesense sensor, the scanning process starts. Once the connection establishment was successful, the text on the "Main Menu" screen is changed to the following 15.

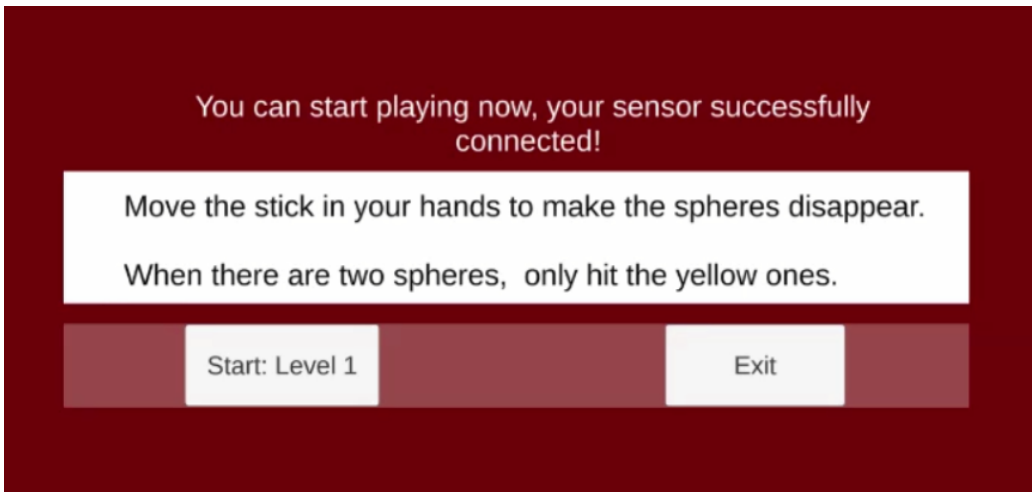


Figure 15: "Main Menu" screen when sensor is connected

[11] MovementGame.apk, available via the link in the appendix B

Using the buttons under the description text leads to the corresponding scenes. When the first level is loaded the screen visualized in figure 16 is shown. The initial position of the stick is described by the angle $\alpha = 0°$.

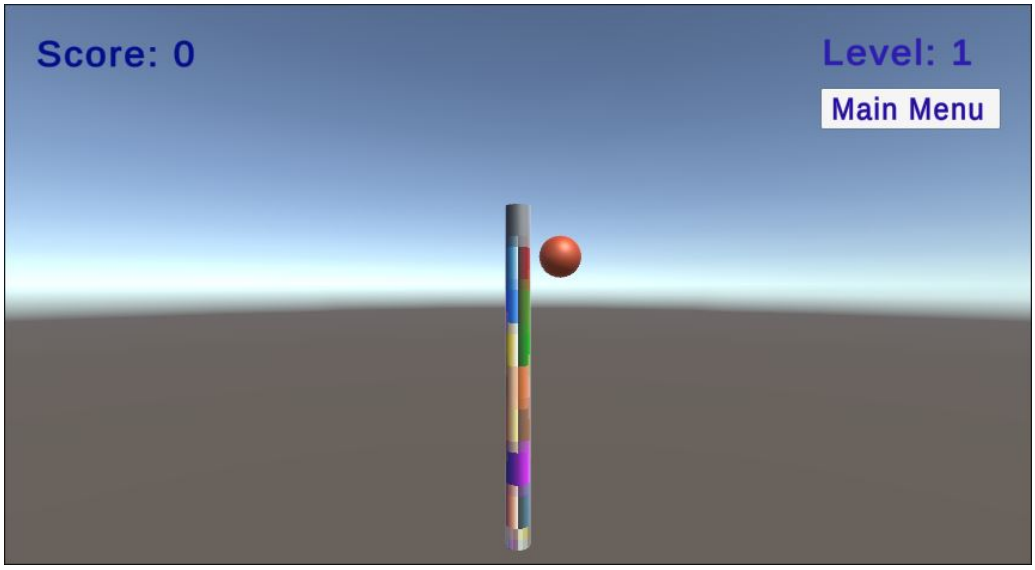


Figure 16: First Level starting position

When the bamboo stick is rotated 90° or further from the initial vertical position to the left or to the right, the stick on the screen stops moving. It never goes under the surface, but a warning message is displayed that informs the player about having moved the stick in the wrong direction. The moment is captured in figure 17. The stick restarts rotating and the warning message is no longer displayed when the performed motion lies in the following angle's range: $-90° < \alpha < 90°$.

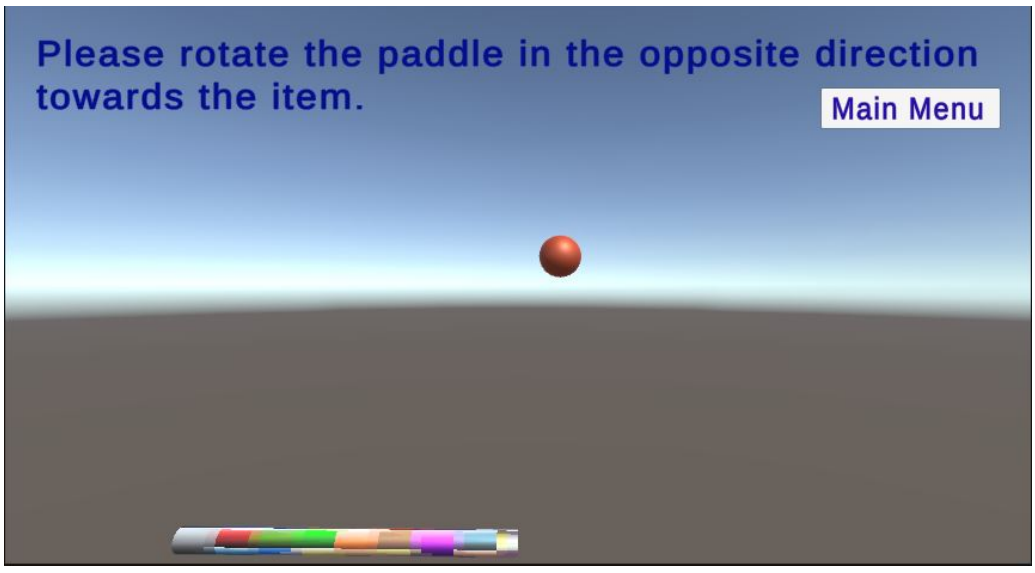


Figure 17: Wrong Movement

When the right movement was performed and the first sphere is hit, it disappears and confetti is displayed. Simultaneously the number of points increases by one. The scene is captured in figure 18. The same scenario counts for the following seven spheres and the following five levels, only differing in the displayed numbers behind the score and level text.

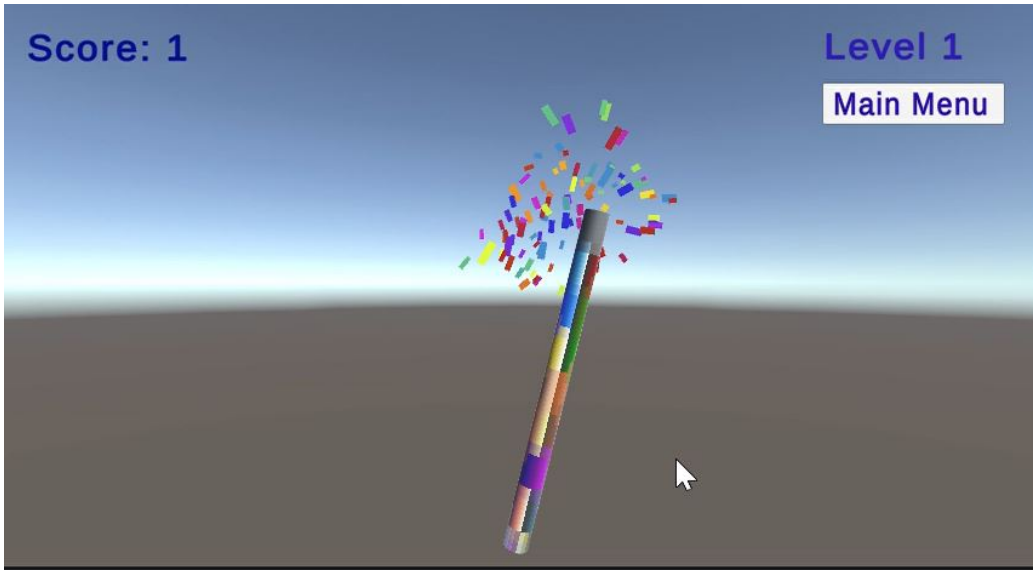


Figure 18: Destruction of sphere and display of confetti

The position of the last spawned sphere in MACS level I is shown in figure 19. If the child cannot perform the movement because it is too difficult, he or she is tired, or for other reason wants to stop the games, the main menu button can be pressed at any time. The game will then be interrupted and return to figure 14, from where the application can be quit by using the "Exit" button.

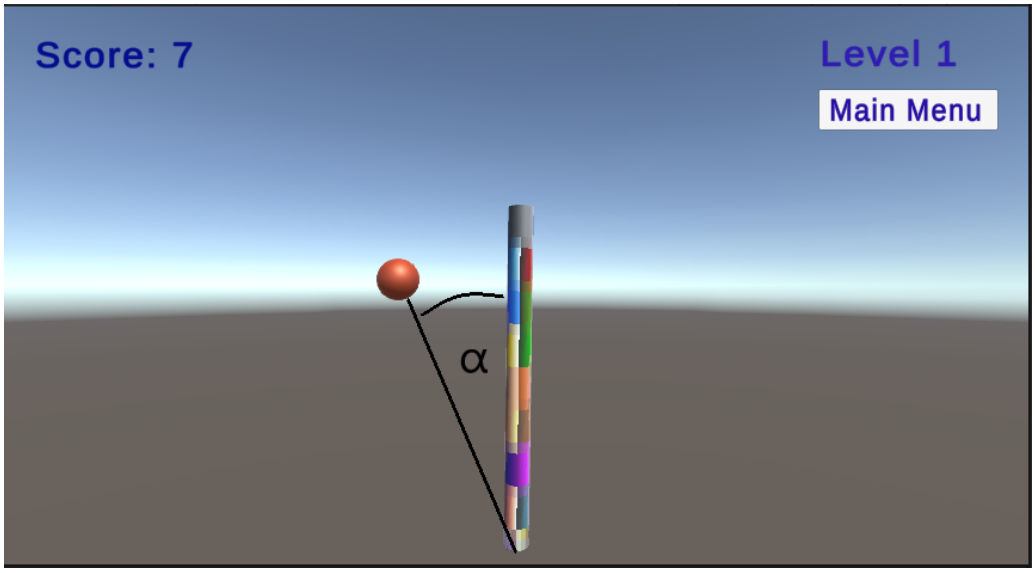


Figure 19: Angle α between stick and last spawned sphere in MACS level III

Once the eight spheres are destroyed one after another, the "Feedback screen" appears. The child is given recognition in the text for its performance and for successfully completing the first level. The screen is visualized in figure 20.

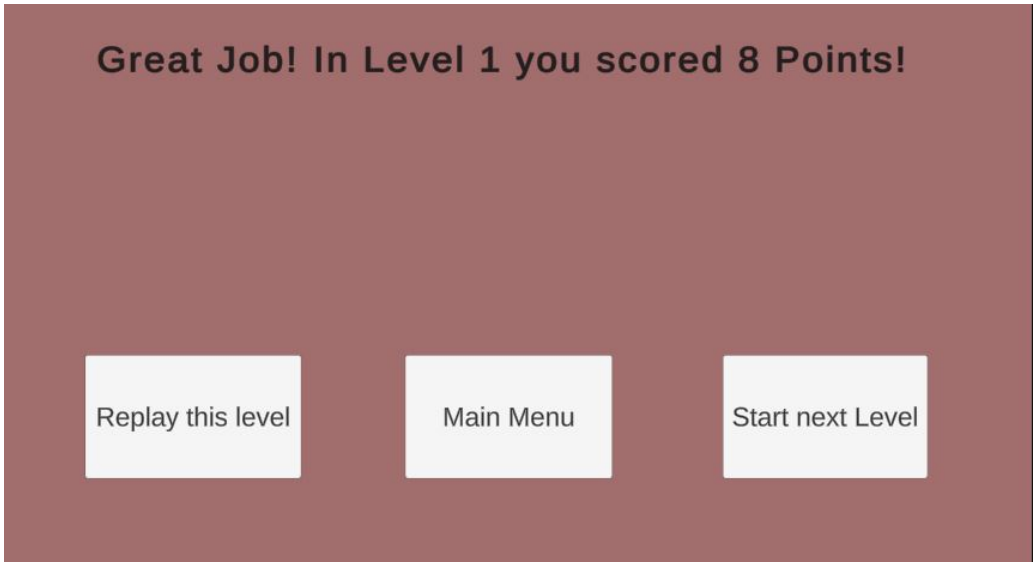


Figure 20: Feedback screen

The three buttons grant the opportunity to return to the main menu 14, to replay the level 16, or to start the next level. By activating the Start next Level button, the second level is loaded. It starts with the following screen 21. The cognitive levels have a different background than the basic ones for visual variety.

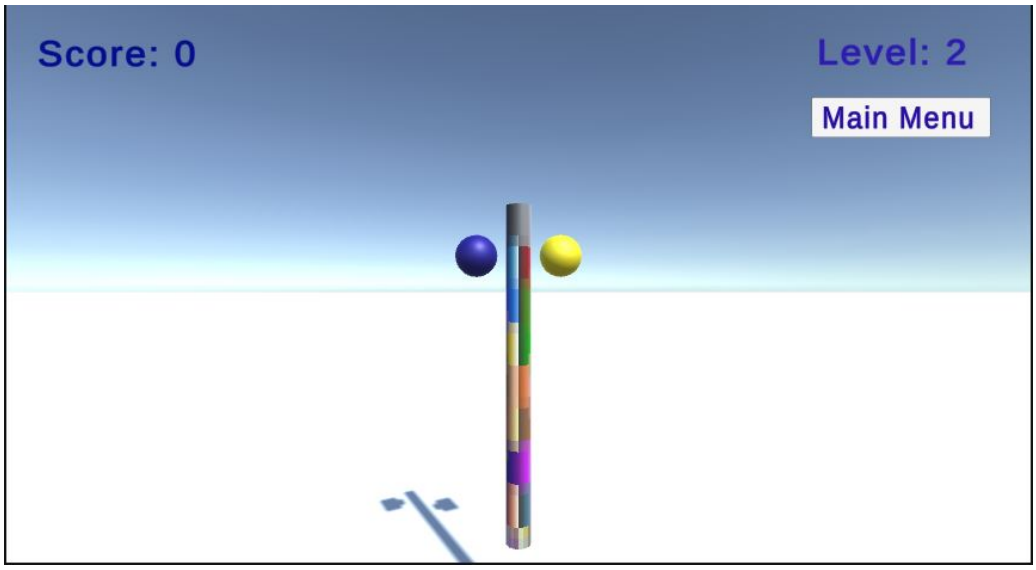


Figure 21: Second Level starting position

The player's task is to hit the yellow sphere. If the blue sphere is hit instead, the following warning message is displayed 22.

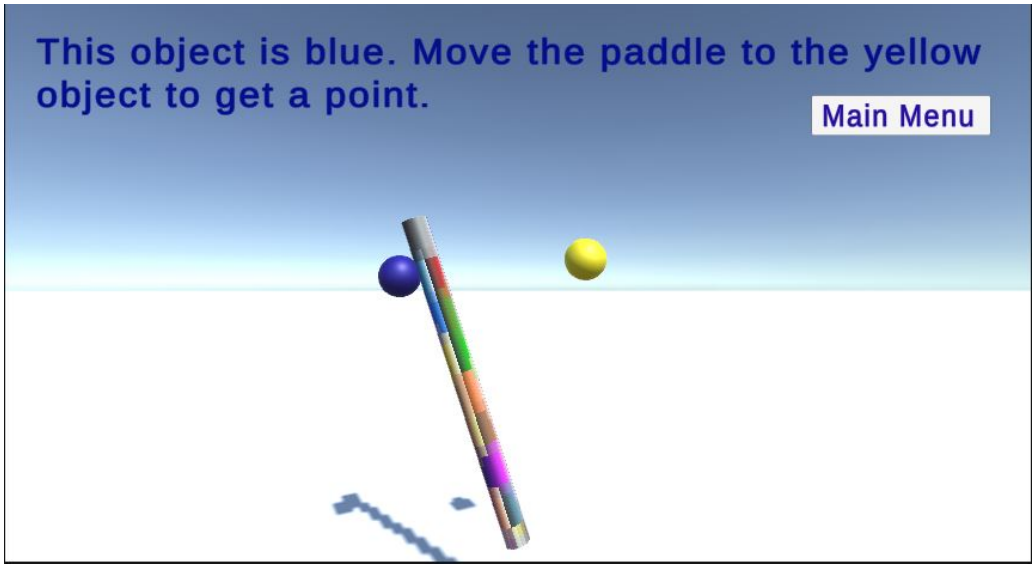
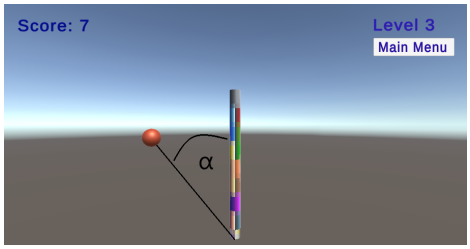
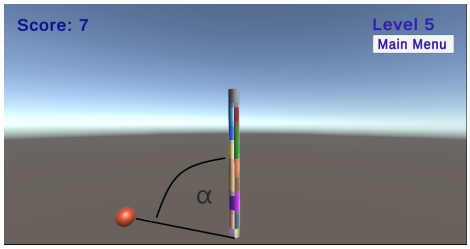


Figure 22: Wrong Item touched

When the yellow sphere is hit, the blue and the yellow sphere both disappear and confetti is displayed at the yellow sphere's position, as described above in figure 18. The then appearing "Feedback screen" looks just like the one in figure 20 with the exchange of the level number. The two presented scenes are both designed for MACS level III. For MACS level II and I, only the angular spans of the spawned spheres differ from these scenarios. Therefore, the two largest angles of the respective levels are additionally captured below 23. The position with the largest angle in MACS level II is shown in figure 23a. For MACS level I the last sphere is spawned at the widest angle. Its position is displayed in figure 23b.



(a) Angle α between stick and last spawned sphere in MACS level II



(b) Angle α between stick and last spawned sphere in MACS level I

Figure 23: Furthest items' positions for higher motor levels

Subsequently, pictures from the rehabilitation system are captured in figure 24. First, the bamboo stick is shown on which the sensor is placed upon. The stick has a length of 270 mm and a diameter of 30 mm. The Movesense sensor has a diameter of $36,6$ mm. On the bottom side of the sensor two screws are located. The bamboo stick's end grain wood was modified to allow the sensor's screws to be mounted on it. Additionally, the word "MOVESENSE" is written on it for the user to be able to determine the alignment of the sensor. The stick's

end grain is visualized in figure 24a. The sensor cannot rotate on the stick and is always aligned in the same direction because of the screws' fix position. The sensor mounted on the bamboo stick is visualised in the second figure 24b. The orientation of the sensor is significant for the motion control of the application. Therefore, multicoloured stickers are attached to the bamboo stick with the aim to ensure the correct orientation of the sensor. The stickers must be facing the player for the correct movement to be executed. A photography is shown in the third figure 24c. The red, black and white sticker on the bottom of the stick mark the area around which the hands are wrapped. In the fourth picture 24d, the moment is captured when a four-year-old child with no motor impairments is playing the game. It serves to demonstrate the inclination of the bamboo stick and the identical inclination of the stick on the screen.

(a) End grain

(b) Sensor applied on stick

(c) Bamboo stick

(d) Child playing the game

Figure 24: Rehabilitation System Movement Game

5 Discussion

In this chapter, the developed system is critically discussed by reviewing the research questions that were addressed in the introduction 1. They are answered in the following:

1. Which game engine is suitable for the implementation of the rehabilitation game?

To choose the SGE for the Movement Game's implementation, the requirements for SGs in a rehabilitation context and the development environment were first considered. The obligatory and optional criteria were elaborated by the developer or defined in exchange with the stakeholder (3.3). Three SGEs were introduced in subsection 2.5. In the preceding SHArKi project the games were developed in GDevelop. This engine was compared to the SGEs Unity and Unreal Engine. These two engines were chosen due to their popularity among developers and their high launching numbers per year [71].
In the benefit analysis 3.3.2, the most suitable engine was determined: Unity Engine 3.3.3. The chosen game engine met all the given requirements. It was especially advantageous because of its plugin for the Movesense sensors, that simplified the development process. For further work the engine can be recommended with some remarks: The Movesense-Plugin is only compatible with Android or Apple devices and cannot run on a laptop [86]. This causes the requirement to build an application whenever the sensors are to be tested. Unity allows to build applications for Android and Apple devices. However an Apple application can only be built if the game was developed on a Macbook [88]. The compatibility of the Movesense-Plugin with the laptop would have simplified the development process. Moreover, it is of great importance to regard the employed Unity version, since the plugin is only supported by an older version 3.3.3. Subsequently it is to be mentioned that Unity's user-interface (UI) led to complications in the work process. The different layer's of the implemented objects were not properly rendered which required the scanning process to be adapted. In the asset's example scenes 3.5.3, a list with the scanned devices is displayed on the screen, to enable the user to select the sensor he or she wants to connect to the application. This could not be replicated. With debug messages it was detected, that the list is printed, however, it was not possible to make it visible on the screen. Therefore, the decision was met to start the scanning automatically, which might be considered advantageous as stated in subsection 3.5.3. At the same time it takes away the possibility of playing alongside other players. It thereby withdraws the potential to socialise with others, which was introduced as design-factor in subsection 2.3. This factor is not taken into account and the game is designed for one person only, as the demanding movement requires a lot of concentration. Furthermore it is presumed, that the young children might be distracted by fel-

low players. However, if the game is played in a room where there are several sensors, this will cause interference with motion control. The application does not know which sensor's motion data to subscribe and the stick therefore does not move as desired. In the future, Unity's complex UI system would need further investigation to avoid this problem and to grant the opportunity to potentially let two players play next to each other. Nevertheless, it is suggested to remove the coin cell batteries from sensors that are not currently in use. Since the sensor's BLE always searches for devices with Bluetooth connection, removing batteries is more energy friendly. Despite the troubles with Unity's UI, the chosen engine was suitable for the rehabilitation game's implementation.

2. Should both hands be trained in the game (bimanual practice) or does it make more sense to focus on the affected side (unimanual practice)?

To answer this question, the unimanual and bimanual therapy method were first reviewed in subsection 2.1.3. Both methods are considered effective in clinical rehabilitation [51]. However, the systematic review about home-based therapy's feasibility and effectiveness [44] stated that only three trials verified the efficacy of HABIT whereas the majority of studies investigated CIMT training. Nevertheless, in the review it is suggested to focus on the coordination of both hands in home-intervention, since it may have a bigger impact on the child's daily life with most activities requiring bimanual hand use [44]. This suggestion is reinforced by Sakzewski's study [28], that indicates that bimanual training may be more efficient in home-based therapy. The decision to train with both hands was following met in accordance with experts, as described in subsection 3.4.2. It aims to enable a wider range of patients with CP to use the training system independently, because the healthy hand can support the affected one. However, the way the system was designed allows each user to decide independently which practice method he or she prefers. Children with less severe courses who can perform supination and pronation movements accurately, might hold the bamboo stick with their affected hand only.

3. How shall a rehabilitation game for children aged $2-4$ years, in which the movements supination and pronation are trained, be designed to be interesting to play, challenging enough for an effective practice, but at the same time not over-demanding?

The Movement Game developed requires the user to supinate or pronate his or her hands to successfully achieve the game's goal to hit the spheres that appear on the screen. The focus of the game is set on the executed motions. As rehabilitation game its purpose is to serve as a medical training system. Whether the concept is sufficient to be perceived as exciting by children with CP aged $2-4$ cannot be answered in this context. However, the attempt to make it exciting

to play was made by considering design principles 2.3. The design of the game meets some of the criteria specified in the frameworks for a successful SG and is critically discussed in the following: Feedback is given after each level. In the application, the feedback text is displayed on the screen. Audible feedback would be beneficial, as the group of players targeted does not know how to read. This also applies to the instructions on how to play the game. Clear and precise rules are given but they are only written on the screen. Optimally, the game would start with a video in which the concept is explained to the children audible and a short performance sequence is shown.

Further suggestions for improvement follow: The application leaks choice and interactivity. This could be changed in further work by offering a selection tool in the "Main Menu". The child is shown a list with pictures of multiple items, for example a basketball, bicycle, fruit, tractors, and emoji. The child can then choose its most favoured item. After the selection, the game starts with the chosen item appearing on the screen, instead of the spheres. After each level, the child is given the option to select another item if he or she has the desire to. That way more variety is brought into the game and a motivational factor for the child is created. Another improvement is the exchange of the played effect when the stick hits the item. Instead of confetti appearing on the screen once the item is triggered, it could shatter into small pieces, a firework could display, or candies might fall out of it. In addition, a visual sound effect could enlarge the astonishment of the child and make the game more exciting to play.

Subsequently, to create a rewarding system, a gift shop could be implemented. Currently, the used stick on the screen is colourfully patterned. In the other conceivable scenario, the stick would be white at the beginning of the game. After each level, the child could either keep playing or visit the gift shop. In the shop, differently priced and sized drawings and patterns are offered. With an empty account the player can still afford the smallest object from the gift shop to prevent demotivation and fatigue. If it has more points it can exchange them with an achievable drawing or pattern. He or she then selects a spot on the stick and the purchased unit is drawn on this position. Whenever the game is replayed by the player, the patterns remain on the stick. If the player exercises regularly and earns enough points to exchange them with drawings and patterns, the stick can eventually be an artwork, painted from top to bottom. This is intended to serve as additional motivation to exercise regularly, but not to withdraw the focus from the movements. Another adjustment could be the implementation of further levels in a similar setting. Thereby a storyline could be drawn and the offered training methods could be extended by focusing on other movements like flexion and extension or abduction and adduction in these levels.

The system currently leaks a monitoring file, as introduced in subsection 2.3 and concerned in subsection 3.4.2. In future, it could be implemented to share rel-

evant informations with the therapists. The data of interest includes the time required to play a level, the ROM and the total amount of executed supination and pronation movements in the training session.

In the following the game's challenge and demand is addressed. The motion control of the stick by the Movesense sensor was realized without jitter, time offset and at the accurate angle. However, hitting the items can be difficult for children with motor restrictions. The player's executed movements must be precise because the stick moves in a three-dimensional environment. For a child with CP, this precision might be challenging. In a two-dimensional space, shattering the objects is simpler because it requires less precision. In addition, the correct alignment of the bamboo stick is mandatory due to its rotational symmetry, but difficult to maintain for young children. Yet the game is a training session with the purpose to make the player execute the movements supination and pronation. When it takes a while for the child to destroy the item with the stick, it still performed supination and pronation movements in the process several times. To facilitate the stick's alignment for the children, a non-symmetrical object could be substituted for the bamboo stick. However, another object is more difficult to be grasped with both hands, which is desired to let the healthy hand support the affected one. Instead, the bamboo stick could be supplemented with a transverse attachment that guides the direction more clearly.
The possibility cannot be ruled out, that the child tries to compensate the intended movements by using his or her upper body. The intention to avoid this is by clearly instructing the child to place his or her forearms and hands on the table. In addition, a supervising person can ensure the correct execution and intervene if necessary. In order to develop the game economically, only a single sensor is used. To avoid the need for supervision, it would be advantageous to have additional sensors attached to the child's upper body to detect compensatory movements. If incorrect movements are detected, a warning message could instruct the child to return to the starting position and to restart the play process.
Three cognitive and three motoric level were implemented to grant the opportunity to not bore and to not over-demand users with more or less severe courses. At this point it is stressed that the selected angles 3.4.3 for varying difficulty are to be understood as exemplary values that can be adjusted if necessary. The angular ranges were derived from literature [4], but still require verification. They could be verified in further work in expert discussions and user-tests. Additionally, some adjustments in the level could be implemented in future. For the motor levels I and II the difficulty could be increased by having a timer set to encourage the player to speed up. When the time has expired and the player hasn't touched the item yet, he or she misses out on receiving a point. Another way to add more speed to the game is by making the items non-stationary bodies. They could

move on a semicircle with lower or higher speed, according to the user's motor level. The child needs to pronate or supinate the stick to catch the moving items. Points could then be given, depending on the time needed to catch the item. Since it is still important that the movement is executed properly, bringing speed into the game again pleads for the attachment of further sensors on the upper body to detect compensatory movements.

Lastly, the efficiency of the system is reviewed. Numerous studies have proven that home programs are efficient in increasing the therapy's intensity [42]. They are furthermore considered effective in motor impairments [51], as already stated in subsection 2.1.3 and discussed in the previous question. The concern that the game has no benefit for the child's motor skills is countered by the highly specific movement. The game is limited to two movements and thus puts the full focus on them. For this work, the system was only tested on healthy adults, as described in the last test cases in subsection 3.5.3. In further steps, analyses might be carried out with affected children in order to test the usability of the system in treatment context. Beforehand, proper user-tests could be examined on healthy adults to further verify the system and to possibly make adjustments. Then, a feasibility study could be conducted to determine if the intervention is likely to be efficacious [14]. Thereby it is examined if further testing e.g. in clinical trials is appropriate [14], to validate whether the movements supination and pronation can be improved by regular use of the Movement Game in addition to physiotherapy.

4. How can the existing system be modified to make it accessible from home and thereby integrable into the children's daily life?

By developing an Android application, it was achieved to make the training system accessible from home. The application can be played from home as long as the user owns a cell phone, tablet, or any other device with an Android operating system. Additionally the user needs a Movesense sensor and a stick to place the sensor on. The system is only complete with these additions and can be used as intended by the developer. The stick needs to be modified in a way that the sensor can be firmly placed and aligned on it. To prevent these work steps, a retainer would be practical, which can be attached to any cylindrical object. Then, for example, an empty cylindrical candy package could be used instead of the bamboo stick for the practice and the child could even be rewarded with some candy after a successful session. The retainer could be designed and 3D printed in further work.
In order to make the game available to a larger group, especially for those who cannot afford the sensor's purchase cost, other adaptations could be considered. At this stage the user inevitably needs a Movesense sensor to use the training system. To avoid the cost, it would be possible to use a second mobile phone

for the movement measurement instead of the sensor. For this adaptation, the framework conditions differ from the developed system and the model therefore would have to be heavily modified. With technical conversions taking place, the game could be controlled with the gyroscope or linear acceleration data of the second cell phone instead of the Movesense sensor. Again, a mechanism would have to be attached to the bamboo stick on which the mobile phone can be fixed. Alternatively another cylindrical object such as a selfie stick could be used to move the motion controller. Lastly, it is to mention that the application could be provided on other target environments. This refers to the operating system on which the game can be hosted and the environment on which it is made available to the stakeholder. In further work, the game could be built as an application running on iOS by using a Macbook and the implemented code from this project. Subsequently, the transfer of the application to the customer's phone could be upgraded. Currently, the application is transmitted to the stakeholder's device via bluetooth or a cable connection. Instead, it could be made available in a cloud or released for the Apple App Store and the Google Play Store.

6 Conclusion

In this work a rehabilitation system for children with CP, aged 2-4, was developed in the form of an Android application. The very specific user group was studied first. Therefore, the disorder CP, the classification system MACS, unimanual and bimanual treatment methods, and the hands' movements supination and pronation were introduced.

Subsequently, SGs that find their application in rehabilitation context and their design requirements were reviewed. Due to the specific requirements of these games, the system's framework conditions were investigated in detail: a suitable game engine was elected and a game concept was established. The three game engines GDevelop, Unity, and Unreal Engine were demonstrated and evaluated in a benefit analysis. Unity, the engine with the highest benefit value, was consequently used for the implementation of the serious game. The game concepts were then elaborated for which a profound investigation of the therapy context was performed. In the chosen concept, the child's task is to make items disappear that are spawned at different angular distances away from a stick, that is shown on the Android device's screen. The child holds a bamboo stick with both hands wrapped around it and performs the movements supination and pronation. The connected IMU, the Movesense sensor, is responsible for the motion control. Via bluetooth conncetion, the sensor's linear acceleration data is sent to the application. It is converted into quaternions and the rotational movement on the screen is generated.

The software development was realized under the V-model's approach. The planned game concept was successfully implemented. Three motor levels were realized, each containing a basic level and a cognitive one. In addition, design requirements such as feedback and motivational factors were incorporated to encourage children to engage in regular exercise sessions. The resulting application works with smooth motion tracking. However, the system was only technically verified and validated. Furthermore, the game was only played by individuals who are not affected by the disorder CP in this setting. In future research could validate the suitability of the Movement Game for the specific application group of children aged $2-4$ with motor restrictions. In this context, both the feasibility of the movements for the children and the motivation to use the system regularly for training purposes are of interest.

Bibliography

Book and Article sources

[1] Leonhard Euler. *Elements of Algebra*. Springer Science & Business Media, 1972. DOI: 10.1007/978-1-4613-8511-0.

[2] Erik B Dam, Martin Koch and Martin Lillholm. "Quaternions, Interpolation and Animation". 1998.

[3] Christine Cans. "Surveillance of cerebral palsy in Europe: a collaboration of cerebral palsy surveys and registers". In: *Developmental Medicine & Child Neurology* 42.12 (2000), pp. 816–824. DOI: https://doi.org/10.1111/j.1469-8749.2000.tb00695.x.

[4] M Kreulen et al. "Three-dimensional video analysis of forearm rotation before and after combined pronator teres rerouting and flexor carpi ulnaris tendon transfer surgery in patients with cerebral palsy". In: *Journal of Hand Surgery* 29.1 (2004), pp. 55–60.

[5] Koman La, BP Smith and JS Shilt. "Cerebral palsy". In: *Lancet* 363.9421 (2004), pp. 1619–31. DOI: https://doi.org/10.1016/S0140-6736(04)16207-7.

[6] K. Himmelmann et al. "The changing panorama of cerebral palsy in Sweden. IX. Prevalence and origin in the birth-year period 1995–1998". In: *Acta Paediatrica* 94.3 (2005), pp. 287–294. DOI: https://doi.org/10.1111/j.1651-2227.2005.tb03071.x.

[7] Ann-Christin Eliasson et al. "The Manual Ability Classification System (MACS) for children with cerebral palsy: scale development and evidence of validity and reliability". In: *Developmental Medicine & Child Neurology* 48.7 (2006), pp. 549–554. DOI: 10.1017/S0012162206001162.

[8] E. Taub et al. "The learned nonuse phenomenon: implications for rehabilitation". In: *Europa Medicophysica* 42.3 (2006), pp. 241–255.

[9] Christine Cans et al. "Recommendations from the SCPE collaborative group for defining and classifying cerebral palsy". In: *Developmental medicine and child neurology* 49.s109 (2007), pp. 35–38.

[10] Heather O Dickinson et al. "Self-reported quality of life of 8–12-year-old children with cerebral palsy: a cross-sectional European study". In: *The Lancet* 369.9580 (2007), pp. 2171–2178. DOI: 10.1016/S0140-6736(07)61013-7.

[11] M. Kreulen et al. “Movement patterns of the upper extremity and trunk associated with impaired forearm rotation in patients with hemiplegic cerebral palsy compared to healthy controls”. In: *Gait & Posture* 25.3 (2007), pp. 485–492. DOI: https://doi.org/10.1016/j.gaitpost.2006.05.015.

[12] Christopher Morris. “Definition and classification of cerebral palsy: a historical perspective”. In: *Developmental Medicine & Child Neurology* 49.s109 (2007), pp. 3–7. DOI: https://doi.org/10.1111/j.1469-8749.2007.tb12609.x.

[13] Peter Rosenbaum et al. “A report: The definition and classification of cerebral palsy April 2006”. In: *Developmental medicine and child neurology. Supplement* 109 (2007), pp. 8–14. DOI: 10.1111/j.1469-8749.2007.tb12610.x.

[14] Deborah J Bowen et al. “How We Design Feasibility Studies”. In: *American journal of preventive medicine* 36.5 (2009), pp. 452–457. DOI: 10.1016/j.amepre.2009.02.002.

[15] Marlene Sandlund, Suzanne McDonough and Charlotte Häger-Ross. “Interactive computer play in rehabilitation of children with sensorimotor disorders: a systematic review”. In: *Developmental Medicine & Child Neurology* 51.3 (2009), pp. 173–179. DOI: https://doi.org/10.1111/j.1469-8749.2008.03184.x.

[16] Andrew M Gordon. “Two hands are better than one: bimanual skill development in children with hemiplegic cerebral palsy”. In: *Developmental Medicine & Child Neurology* 52.4 (2010), pp. 315–316. DOI: https://doi.org/10.1111/j.1469-8749.2009.03390.x.

[17] Ellen Jaspers et al. “Three-dimensional upper limb movement characteristics in children with hemiplegic cerebral palsy and typically developing children”. In: *Research in Developmental Disabilities* 32.6 (2011), pp. 2283–2294. DOI: https://doi.org/10.1016/j.ridd.2011.07.038.

[18] Michael Livingston et al. “Exploring Issues of Participation Among Adolescents with Cerebral Palsy: What’s Important to Them?” In: *Physical & occupational therapy in pediatrics* 31 (2011), pp. 275–287. DOI: https://doi.org/10.3109/01942638.2011.565866.

[19] L. Sakzewski et al. “Impact of intensive upper limb rehabilitation on quality of life: a randomized trial in children with unilateral cerebral palsy”. In: *Developmental Medicine & Child Neurology* 54.5 (2012), pp. 415–423. DOI: https://doi.org/10.1111/j.1469-8749.2012.04272.x.

[20] Marije de Bruin et al. “Biceps brachii can add to performance of tasks requiring supination in cerebral palsy patients”. In: *Journal of Electromyography and Kinesiology* 23.2 (2013), pp. 516–522. DOI: https://doi.org/10.1016/j.jelekin.2012.10.013.

[21] Ermellina Fedrizzi et al. “Unimanual and Bimanual Intensive Training in Children With Hemiplegic Cerebral Palsy and Persistence in Time of Hand Function Improvement: 6-Month Follow-Up Results of a Multisite Clinical Trial”. In: *Journal of Child Neurology* 28.2 (2013), pp. 161–175. DOI: 10.1177/0883073812443004.

[22] “ISO/IEC/IEEE International Standard - Software and systems engineering — Software testing —Part 3: Test documentation”. In: *ISO/IEC/IEEE 29119-3:2021(E)* (2013), pp. 1–98.

[23] Keith Lohse et al. “Video games and rehabilitation: using design principles to enhance engagement in physical therapy”. In: *Journal of Neurologic Physical Therapy* 37.4 (2013), pp. 166–175. DOI: 10.1097/NPT.0000000000000017.

[24] Iona Novak et al. “A systematic review of interventions for children with cerebral palsy: state of the evidence”. In: *Developmental medicine & child neurology* 55.10 (2013), pp. 885–910. DOI: 10.1111/dmcn.12246.

[25] Gerhard Pahl et al. *Pahl/Beitz Konstruktionslehre: Grundlagen erfolgreicher Produktentwicklung. Methoden und Anwendung.* Springer-Verlag, 2013.

[26] B. Bonnechère et al. “Can serious games be incorporated with conventional treatment of children with cerebral palsy? A review”. In: *Research in Developmental Disabilities* 35.8 (2014). DOI: https://doi.org/10.1016/j.ridd.2014.04.016.

[27] Antoni Jaume-i-Capó, Biel Moyà-Alcover and Javier Varona. “Design issues for vision-based motor-rehabilitation serious games”. In: *Technologies of inclusive well-being*. Springer, 2014, pp. 13–24. DOI: 10.1007/978-3-642-45432-5_2.

[28] Leanne Sakzewski, Jenny Ziviani and Roslyn N Boyd. “Efficacy of Upper Limb Therapies for Unilateral Cerebral Palsy: A Meta-analysis”. In: *Pediatrics* 133.1 (2014), e175–e204. DOI: 10.1542/peds.2013-0675.

[29] Alexander Uskov and Bhuvana Sekar. “Serious games, gamification and game engines to support framework activities in engineering: Case studies, analysis, classifications and outcomes”. In: *IEEE International Conference on Electro/Information Technology*. 2014, pp. 618–623. DOI: 10.1109/EIT.2014.6871836.

[30] Christof Zangemeister. *Nutzwertanalyse in der Systemtechnik: eine Methodik zur multidimensionalen Bewertung und Auswahl von Projektalternativen*. Winnemark, Germany, 2014.

[31] L. Sakzewski, N. Branjerdporn and R. Boyd. "Individual patient data meta-analysis of intensive upper limb therapy approaches on upper limb functional outcomes for children with unilateral cerebral palsy". In: *Developmental Medicine & Child Neurology* 57.S5 (2015), pp. 80–81. DOI: https://doi.org/10.1111/dmcn.9_12886.

[32] Roberto Valenti, Ivan Dryanovski and Jizhong Xiao. "Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs". In: *Sensors* 15 (Aug. 2015), pp. 19302–19330. DOI: 10.3390/s150819302.

[33] Yong Yu et al. "Production of effective stretch reflex by a pronation and supination function recovery training device for hemiplegic forearms". In: *IEEE International Conference on Robotics and Biomimetics (ROBIO)* (2015), pp. 150–157. DOI: https://doi.org/10.1109/ROBIO.2015.7418759.

[34] Zulfiqar Ali and Muhammad Usman. "A framework for game engine selection for gamification and serious games". In: *2016 Future Technologies Conference (FTC)*. 2016, pp. 1199–1207. DOI: 10.1109/FTC.2016.7821753.

[35] H Kerr Graham et al. "Cerebral palsy". In: *Nature Reviews Disease Primers* 2.15082 (2016), pp. 1–1. DOI: https://doi.org/10.1038/nrdp.2015.82.

[36] Marco Iosa et al. "Wearable inertial sensors for human movement analysis". In: *Expert Review of Medical Devices* 13.7 (2016), pp. 641–659. DOI: 10.1080/17434440.2016.1198694.

[37] Angela Shierk, Amy Lake and Tara Haas. "Review of Therapeutic Interventions for the Upper Limb Classified by Manual Ability in Children with Cerebral Palsy". In: *Seminars in plastic surgery* 30.1 (2016), pp. 14–23. DOI: 10.1055/s-0035-1571256.

[38] Marc Soubeyrand et al. "Pronation and supination of the hand: Anatomy and biomechanics." In: *Hand surgery & rehabilitation* 36 1 (2016), pp. 2–11. DOI: https://doi.org/10.1016/j.hansur.2016.09.012.

[39] Amengual E. Alcover, Antoni Jaume-I-CapCapó and Gabriel Moyà-Alcover. "PROGame: A process framework for serious game development for motor rehabilitation therapy". In: *PLOS ONE* 13 (2018), e0197383. DOI: 10.1371/journal.pone.0197383.

[40] Daniel Angermeier et al. *V-Modell XT. Das deutsche Referenzmodell für Systementwicklungsprojekte. Version: 2.3*. Verein zur Weiterentwicklung des V-Modell XT e.V. (Weit e.V.), 2018. URL: https://www.cio.bund.de/Webs/CIO/DE/digitaler-wandel/Achitekturen_und_Standards/V_modell_xt/v_modell_xt_links_und_downloads/v_modell_xt_links_und_downloads_node.html.

[41] David A Hobbs et al. "A Custom Serious Games System with Forced-Bimanual Use can Improve Upper Limb Function for Children with Cerebral Palsy: Results from a Randomised Controlled Trial". In: *Technology and Disability*. Vol. 31. IOS PRESS, July 2019, S147–S148. DOI: 10.3233/TAD-190009.

[42] Iona Novak and Ingrid Honan. "Effectiveness of paediatric occupational therapy for children with disabilities: A systematic review". In: *Australian occupational therapy journal* 66.3 (2019), pp. 258–273. DOI: 10.1111/1440-1630.12573.

[43] "An Algorithm for Quaternion–Based 3D Rotation". In: *International Journal of Applied Mathematics and Computer Science* 30.1 (2020), pp. 149–160. DOI: 10.34768/AMCS-2020-0012.

[44] Laura W M E Beckers et al. "Feasibility and effectiveness of home-based therapy programmes for children with cerebral palsy: a systematic review". In: *BMJ Open* 10.10 (2020). DOI: 10.1136/bmjopen-2019-035454.

[45] Cristina Carmona-Pérez et al. "Concurrent Validity and Reliability of an Inertial Measurement Unit for the Assessment of Craniocervical Range of Motion in Subjects with Cerebral Palsy". In: *Diagnostics* 10 (Feb. 2020), p. 80. DOI: 10.3390/diagnostics10020080.

[46] Mayra Carrión-Toro et al. "iPlus a user-centered methodology for serious games design". In: *Applied Sciences* 10.24 (2020), p. 9007. DOI: 10.3390/app10249007.

[47] Eric M. Chin et al. "Principles of Medical and Surgical Treatment of Cerebral Palsy". In: *Neurologic Clinics* 38.2 (2020), pp. 397–416. DOI: https://doi.org/10.1016/j.ncl.2020.01.009.

[48] Iris Graessler and Julian Hentze. "The new V-Model of VDI 2206 and its validation". In: *at - Automatisierungstechnik* 68.5 (2020), pp. 312–324. DOI: doi:10.1515/auto-2020-0015.

[49] Afzal Hussain et al. "Unity Game Development Engine: A Technical Survey". In: *University of Sindh Journal of Information and Communication Technology* 4 (2020).

[50] Christina Mittag et al. “Development of a home-based wrist range-of-motion training system for children with cerebral palsy”. In: *at - Automatisierungstechnik* 68.11 (2020), pp. 967–977. DOI: doi:10.1515/auto-2020-0085.

[51] Iona Novak et al. “State of the Evidence Traffic Lights 2019: Systematic Review of Interventions for Preventing and Treating Children with Cerebral Palsy”. In: *Current neurology and neuroscience reports* 20.3 (2020), pp. 1–21. DOI: https://doi.org/10.1007/s11910-020-1022-z.

[52] Johanna van Schaik and Nadia Dominici. “Motion tracking in developmental research: Methods, considerations, and applications”. In: *Progress in Brain Research* 254 (July 2020). DOI: 10.1016/bs.pbr.2020.06.007.

[53] Celia Francisco-Martínez et al. “Upper Limb Movement Measurement Systems for Cerebral Palsy: A Systematic Literature Review”. In: *Sensors* 21 (Nov. 2021), pp. 1–17. DOI: 10.3390/s21237884.

[54] Irene Alice Chicchi Giglioli et al. “Are 3D virtual environments better than 2D interfaces in serious games performance? An explorative study for the assessment of executive functions”. In: *Applied Neuropsychology: Adult* 28.2 (2021), pp. 148–157. DOI: 10.1080/23279095.2019.1607735.

[55] Verein deutscher Ingenieure. *VDI/VDE 2206: Development of mechatronic and cyber-physical systems*. 2021.

[56] Siavash Khaksar et al. “Application of Inertial Measurement Units and Machine Learning Classification in Cerebral Palsy”. In: *JMIR Rehabilitation and Assistive Technologies* 8 (Apr. 2021). DOI: 10.2196/29769.

[57] John Voight. “Quaternion Algebras”. In: Graduate Texts in Mathematics (Jan. 2021). DOI: 10.1007/978-3-030-56694-4.

[58] W. Coenen et al. “Manuelle Medizin: Therapie der Wahl bei infantiler Zerebralparese”. In: *Manuelle Medizin* 60 (Oct. 2022). DOI: 10.1007/s00337-022-00912-z.

[59] Kayley Crebbin et al. “The Use of Serious Gaming to Improve Sensorimotor Function and Motivation in People with Cerebral Palsy: A Systematic Review”. In: *Games for Health Journal* 11.6 (2022). DOI: 10.1089/g4h.2022.0112.

[60] Psiche Giannoni and Liliana Zerbino. *Cerebral Palsy: A Practical Guide for Rehabilitation Professionals*. Springer, 2022. DOI: https://doi.org/10.1007/978-3-030-85619-9.

[61] Andrew M Gordon et al. “HABIT+tDCS: a study protocol of a randomised controlled trial (RCT) investigating the synergistic efficacy of hand-arm bimanual intensive therapy (HABIT) plus targeted non-invasive brain stimulation to improve upper extremity function in school-age children with unilateral cerebral palsy”. In: *BMJ Open* 12.2 (2022). DOI: 10.1136/bmjopen-2021-052409.

[62] Daniel Laidig and Thomas Seel. “VQF: Highly Accurate IMU Orientation Estimation with Bias Estimation and Magnetic Disturbance Rejection”. In: (2022). DOI: 10.48550/ARXIV.2203.17024.

[63] Karsten Lehn, Merijam Gotzes and Frank Klawonn. “Grundlagen der Computergrafik: Eine Einführung mit OpenGL und Java”. In: (2022). DOI: 10.1007/978-3-658-36075-7_5.

[64] Edmund Weitz. “Die Antike und das Mittelalter”. In: *Gesichter der Mathematik: 111 Porträts und biographische Miniaturen*. Springer Berlin Heidelberg, 2022, pp. 1–16. DOI: 10.1007/978-3-662-66349-3_1.

Online sources

[65] *Android Logcat Guide*. URL: https://docs.unity3d.com/Packages/com.unity.mobile.android-logcat@0.1/manual/index.html (visited on 12/1/2023).

[66] *Codieren Sie schneller. Arbeiten Sie intelligenter. Gestalten Sie die Zukunft mit Visual Studio 2022*. URL: https://visualstudio.microsoft.com/de/vs/ (visited on 12/1/2023).

[67] *Creating Games for Android*. URL: https://unity.com/solutions/mobile/android-game-development (visited on 12/1/2023).

[68] *Flowcharts in Programming Applications & Best Practices*. URL: https://www.zenflowchart.com/guides/flowcharts-in-programming (visited on 10/1/2023).

[69] *Forums for Unreal Engine & MetaHuman*. URL: https://forums.unrealengine.com/ (visited on 8/1/2023).

[70] *Futternapf-Set Edelstahl 2x 400ml mit Silikonmatte Grau*. URL: https://huskl.de/product/futternapf-set-2x400ml-silikonmatte-grau/ (visited on 9/1/2023).

[71] *Game engines on Steam: The definitive breakdown*. URL: https://www.gamedeveloper.com/business/game-engines-on-steam-the-definitive-breakdown (visited on 13/9/2022).

[72] *GDevelop*. URL: https://gdevelop.io/ (visited on 18/11/2022).

[73] *GDevelop Wiki Levels*. URL: https://wiki.gdevelop.io/gdevelop5/tutorials/space-shooter/12-levels (visited on 8/1/2023).

[74] *GDevelop Wiki Resources*. URL: https://wiki.gdevelop.io/gdevelop5/tutorials/resources (visited on 8/1/2023).

[75] *GDevelop: Welche Spiele-Engine ist am besten für Anfänger?* URL: https://gdevelop.io/de-de/page/which-game-engine-is-best-for-beginners (visited on 18/11/2022).

[76] *Join The GDevelop Community*. URL: https://gdevelop.io/page/community (visited on 8/1/2023).

[77] *Movesense*. URL: https://www.movesense.com/ (visited on 18/11/2022).

[78] *Movesense: BLE Communication*. URL: https://www.movesense.com/docs/system/power_consumption/#ble-communication (visited on 15/1/2023).

[79] *Schmetterlingsnetz weiss 40 cm - aufklappbarer Bügel*. URL: https://www.vivara.de/schmetterlingsnetz-weiss-40-cm-aufklappbarer-bugel (visited on 9/1/2023).

[80] *Suunto: Movesense*. URL: https://www.suunto.com/de-de/Campaigns/Movesense/ (visited on 15/1/2023).

[81] *Twitter: GDevelop*. URL: https://mobile.twitter.com/gdevelopapp/with_replies (visited on 11/1/2023).

[82] *Twitter: Unity Labs*. URL: https://mobile.twitter.com/unity3dlabs (visited on 11/1/2023).

[83] *Twitter: Unreal Engine*. URL: https://mobile.twitter.com/unrealengine (visited on 11/1/2023).

[84] *Unity 2019.4.40*. URL: https://unity.com/releases/editor/whats-new/2019.4.40%5C#release-urldates (visited on 12/1/2023).

[85] *Unity Asset Store*. URL: https://assetstore.unity.com/?category=3d&platform=standalonelinuxuniversal&free=true&orderBy=1 (visited on 20/9/2021).

[86] *Unity Asset Store: Movesense Sensor Plugin*. URL: https://assetstore.unity.com/packages/tools/integration/movesense-sensor-plugin-118242#description (visited on 12/1/2023).

[87] *Unity Build immersive VR experiences*. URL: https://unity.com/unity/features/vr (visited on 18/11/2022).

[88] *Unity Documentation: Building for iOS*. URL: https://docs.unity3d.com/Manual/UnityCloudBuildiOS.html (visited on 25/1/2023).

[89] *Unity Documentation: Collider.OnTriggerEnter(Collider)*. URL: https://docs.unity3d.com/ScriptReference/Collider.OnTriggerEnter.html (visited on 20/1/2023).

[90] *Unity Documentation: Important Classes - Quaternion*. URL: https://docs.unity3d.com/Manual/class-Quaternion.html (visited on 19/12/2022).

[91] *Unity Documentation: Quaternion.AngleAxis*. URL: https://docs.unity3d.com/ScriptReference/Quaternion.AngleAxis.html (visited on 20/1/2023).

[92] *Unity Documentation: Quaternion.Normalize*. URL: https://docs.unity3d.com/ScriptReference/Quaternion.Normalize.html (visited on 20/1/2023).

[93] *Unity Documentation: Quaternion.ToAngleAxis*. URL: https://docs.unity3d.com/ScriptReference/Quaternion.ToAngleAxis.html (visited on 20/1/2023).

[94] *Unity Documentation: Rotation and orientation in Unity*. URL: https://docs.unity3d.com/Manual/QuaternionAndEulerRotationsInUnity.html (visited on 19/12/2022).

[95] *Unity Documentation: Transform.rotation*. URL: https://docs.unity3d.com/2021.2/Documentation/ScriptReference/Transform-rotation.html (visited on 20/1/2023).

[96] *Unity download archive*. URL: https://unity.com/releases/editor/archive (visited on 12/1/2023).

[97] *Unity Gaming Solution*. URL: https://unity.com/solutions/game (visited on 18/11/2022).

[98] *Unity: Programming in Unity*. URL: https://unity.com/solutions/programming (visited on 27/9/2022).

[99] *Unreal Engine*. URL: https://www.unrealengine.com/en-US (visited on 27/9/2022).

[100] *Unreal Engine 5 Android Quick Start*. URL: https://docs.unrealengine.com/5.1/en-US/setting-up-unreal-engine-projects-for-android-development/ (visited on 12/1/2023).

[101] *Unreal Engine Assetshop*. URL: https://unrealengine.com/marketplace/en-US/store (visited on 27/9/2022).

[102] *Unreal Engine Community Wiki: Unreal vs Unity*. URL: https://unrealcommunity.wiki/differences-between-unity-and-unreal-b2c4rqwm (visited on 27/9/2022).

[103] *Unreal Engine FAQ*. URL: https://www.unrealengine.com/en-US/faq (visited on 27/9/2022).

[104] *Unreal Engine Learn*. URL: https://www.unrealengine.com/en-US/learn (visited on 27/9/2022).

[105] *Unreal Engine Marketplace*. URL: https://unrealengine.com/marketplace/en-US/store (visited on 8/1/2023).

[106] *VDI/VDE 2206 "Entwicklung mechatronischer und cyber-physischer Systeme"*. URL: https://www.vdi.de/richtlinien/programme-zu-vdi-richtlinien/vdi-2206 (visited on 12/9/2022).

[107] *Wooden Story: Sortierbox Holz*. URL: https://www.littlegreenie.de/sortierbox-holz-1071?sPartner=100100&number=WSN19WS20&gclid=CjwKCAiAk--dBhABEiwAchIwkWPzVhM61zS8cJnSNKFr-Z0iUupQkQy4ODcXXmh8KCcIYZQTAMhy1RoCIUYQAvD_BwE (visited on 9/1/2023).

Appendix

Appendix A Mini-MACS

Information for users

The Mini-Manual Ability Classification System (Mini-MACS) is a classification system that describes how children with cerebral palsy (CP) aged 1–4 years use their hands when handling objects in daily activities. Ability is ranked on five levels based on the children's self-initiated ability and their need for assistance or adaptation when handling objects. This brochure also describes differences between adjacent levels to make it easier to determine the most appropriate level. Mini-MACS is a functional description that can be used as a complement to the supposed diagnose of CP and its subtypes.

The description concern how the children handle objects relevant for age. The objects referred to are those commonly found in the children's environment which they use when performing tasks, such as playing, drawing, eating, or dressing. How children handle toys often gives a good idea of their manual ability. Obviously, a 12-month-old child does not handle the same toys and other objects as a 4-year-old. A child's motivation and cognitive ability also influence the ability to handle objects and, consequently, the Mini-MACS level.

When assessing a child's Mini-MACS level, choose the level that best describes the child's usual performance in the daily environment. To better understand what a child usually does, and how he or she performs this activity, it is necessary to ask someone who knows the child well. The questions should be phrased to obtain a description of the type of objects the child handles, in what situations, and how. Mini-MACS levels reflect what the child usually does, not his or her best performance as demonstrated in a specific test situation.

Mini-MACS assesses the child's general ability to handle everyday objects, not the function of each hand separately. Mini-MACS does not intend to explain the underlying reasons for impaired manual capacity.

The Mini-MACS system spans the entire spectrum of functional limitation found among children with CP and covers all CP sub diagnoses. Level I includes children with minor limitations, if any, while children with severe disabilities are usually classified on level V. Certain CP subtypes can be found at all levels, e.g., bilateral CP, while unilateral CP usually occurs at levels I–III. Mini-MACS does not include children without physical disabilities; if it did, they would be classified as level "0". However, no such level exists!

Since Mini-MACS consists of only five levels, each level includes children with relatively varied function. Consequently, Mini-MACS is a classification system, probably not sensitive to changes and should therefore not be used to evaluate development or interventions. Mini-MACS can be used to describe and differentiate into five levels functional aspects on how a suspected CP diagnosis affects children's manual ability.

The five-level Mini-MACS scale is ordinal, which means that the differences between levels are not necessarily equal, nor are children with CP equally distributed across the five levels.

Contributers: Ann-Marie Öhrwall, Ulla Wahlström, Åsa Persson-Annersten, E-mail: Ann-Christin.Eliasson@ki.se
www.macs.nu

Mini-Manual Ability Classification System for children with cerebral palsy 1 - 4 years of age

The Manual Ability Classification System (MACS) described how children aged 4–18 years with CP use their hands when handling objects in daily activities. Mini-MACS is an adaptation of MACS for children aged 1–4 years.

- Mini-MACS classifies children's ability to handle objects that are relevant for their age and development as well as their need for support and assistance in such situations.
- Mini-MACS describes how children usually use their hands to handle objects, such as toys, in various settings. In other words, it describes what they ordinarily do, rather than what is known to be their best capacity.
- Mini-MACS classifies the child's overall ability to handle objects, not the ability of each hand separately.
- To find out how a child handles various objects in everyday life, it is necessary to ask someone who knows the child well. Such knowledge cannot be obtained through specific testing. The questions should be phrased to obtain a description of the type of objects the child handles daily, in what situations, and how.

Mini-MACS

What do you need to know to use Mini-MACS?

Mini-MACS users need to find out what objects the child usually handles and how they handle them: with ease or difficulty, quickly or slowly, with precision or randomly? For example, you can ask about and/or observe how the child uses his or her hands when playing and during meals, or when participating in usual activities of daily living.

Ask questions about the child's self-initiated ability and how much adult help and support the child needs to handle everyday objects, e.g. toys.

Below is a description of the five Mini-MACS levels of children's self-initiated ability and their need for assistance or adaptation when handling objects.

I. **Handles objects easily and successfully**. The child may have a slight limitation in performing actions that require precision and coordination between the hands but they can still perform them. The child may need somewhat more adult assistance when handling objects compared to other children of the same age.

II. **Handles most objects, but with somewhat reduced quality and/or speed of achievement.** Some actions can only be performed and accomplished with some difficulty and after practice. The child may try an alternative approach, such as using only one hand. The child need adult assistance to handle objects more frequently compared to children at the same age.

III. **Handles objects with difficulty.** Performance is slow, with limited variation and quality. Easily managed objects are handled independently for short periods. The child often needs adult help and support to handle objects.

IV. **Handles a limited selection of easily managed objects in simple actions.** The actions are performed slowly, with exertion and/or random precision. The child needs constant adult help and support to handle objects.

V. **Does not handle objects and has severely limited ability to perform even simple actions.** At best, the child can push, touch, press, or hold on to a few items, in constant interaction with an adult.

Distinctions between Levels I and II

Children in Level I may have slightly more difficulty handling items that require good fine motor skills compared to children without disabilities of the same age.

Children in Level II handle essentially the same objects as children in Level I, but they may encounter problems performing tasks and/or take longer to perform them, so they often ask for help. Functional differences between hands may cause performance to be less effective. They may need more guidance and practice to learn how to handle objects compared with children in Level I.

Distinctions between Levels II and III

Children in Level II can handle most objects, though they may take longer and do so with somewhat less quality, and they may need a lot of guidance and practice to learn how to handle objects.

Level III children manage to use easily handled objects but often need help placing objects in an easy position in front of them. They perform actions with few subcomponents. Performance is slow.

Distinctions between Levels III and IV

Children in Level III manage to use easily handled objects independently for short periods. They perform actions with few subcomponents, and the actions take a long time to perform.

At best, children in Level IV can perform simple actions such as grasping and releasing easily handled objects that are offered in an adapted position. They need constant help.

Distinctions between Levels IV and V

Children in Level IV perform individual actions with a very limited selection of objects and need constant help.

At best, children in Level V perform simple movements in special situations. For example, they can press a simple button or hold single, simple objects.

Appendix B Movement Game

https://tubcloud.tu-berlin.de/s/zzz53NqWP7rCAPs

Appendix C Movesense-Plugin

#	File or Source	Action
1	[86]	Download Movesense-Plugin
2	Unity	Window<Package manager<My Assets<Movesense Sensor Plugin<Import
3	Unity	Project<Assets<Movesense-Plugin<Documentation Movesenseplugin V4
4	Unity	Project<Assets<Movesense-Plugin<Scripts<Editor<MDS Updater<Collections<MDSxConstants
5	MDSxConstants.cs	Exchange word ”movesense” with ”suunto”
6	Unity	Tools<Movesense Library<Update

Appendix D Movesense Sensor HR+

MOVESENSE SENSOR HR+

Versatile, light and small but extremely durable sensor capable of measuring any movement and much more. Customizable functionality through open APIs that enable development of unique in-device apps. The functionality can be tailored to fit the exact needs of the target use case.

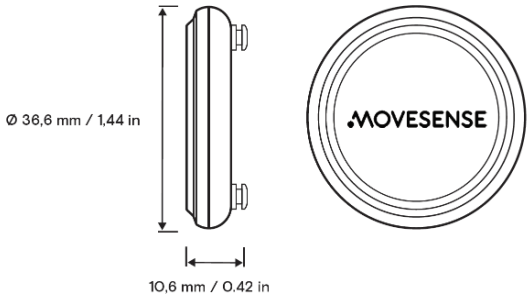
- Swim and shock proof construction, suitable for any sports and other activities
- Low profile snap connection for smooth and subtle attachment to apparel or gear
- User replaceable coin cell battery
- State of the art ultra-low power components
- Small size, light weight and waterproof
- Based on Suunto design and development
- Developed, designed and manufactured in Finland
- Available with custom branding

TECHNICAL HIGHLIGHTS

- 9-axis motion sensor: acceleration, gyroscope, magnetometer
- Heart rate, R-R- intervals, BLE heart rate service, optional: single channel ECG
- 1-wire expansion bus
- Temperature
- Data logging memory
- Bluetooth® 4.0/5.0 radio depending on firmware version
- Tools for developing customized applications that run on the sensor
- Software libraries for developing compatible mobile applications
- Wireless firmware update capability
- Recognizes its attachment base through unique ID in Movesense connector

MOVESENSE

TECHNICAL BRIEF



DIMENSIONS

- 36.6mm/1.44” dia. x 10.6mm/0.42” thick
- Weight 9.4g / 0.34oz with battery
- Water resistant to 30m/100ft

I/O

- Red led on the front, SW controllable
- Wake-up, heart rate and 1-wire expansion
- Interface via Movesense studs, stud center-to-center distance: 27.0mm/1.06”
- Optional Smart Attachments with unique ID (readable through 1-wire)

SENSORS

- Acceleration & Gyroscope
  - ±2/±4/±8/±16g full scale ±125/±245/±500/±1000/±2000°/s full scale, sampling frequency: 12.5/26/52/104/208/416/833Hz
- Magnetometer ±49 gauss full scale
- Temperature
  - accuracy <±0.5°C, 0°C to +65°C
- Heart rate
  - Beats/minute, RR intervals, BLE HR service
  - 1-Channel ECG (non-medical)

LOGGER MEMORY

- 3Mbit EEPROM

SOFTWARE

- SDK for developing apps for the sensor
- Sensors and peripherals controllable via API • incl. BT advertising, power schemes
- Easy to use C++ Movesense Device API
- iOS and Android mobile libraries with wireless sensor firmware update capability
- GNU toolchain for embedded ARM

MCU

- Nordic Semiconductor nRF52832
- 32 –bit ARM® Cortex®-M4
- 64kB on-chip RAM*
- 512kB on-chip FLASH*

  (*) Memory is shared between the Movesense Os and the user application
- Bluetooth Low Energy radio

BATTERY

- CR 2025 Lithium coin cell battery
- Operating time up to months, depending on the user application

APPROVALS AND COMPLIANCES

- CE, FCC, IC, C-Tick, CMIIT
- Conforms REACH, RoHS
- Bluetooth 4.0 / 5.0

PATENTS

US 13/071,624, US 13/832,049, US 13/832,598, US 13/917,668, US 13/397,872, USD 667,127, US 8,386,009, US 8,750,959, US 8,814,574, US 8,886,281, others pending.