AUGMENTING PATIENT THERAPIES WITH VIDEO GAME TECHNOLOGY

Thesis by

Richard Davison

School of Computing Science

In Partial Fulfilment of the Requirements

for the Degree of

Doctor of Philosophy

Newcastle University

Newcastle-Upon-Tyne, UK

September 23, 2016

Dedication

This thesis is dedicated to my grandparents, who always inspired me to pursue a future in academia.

Acknowledgements

I am immensely grateful to my family and my partner Ana for the support throughout my studies. I would also like to thank the many friends I met during the course of my studies, whose encouragement aided me immensely over the years of study leading up to this thesis.

I would like to thank Graham Morgan and Gary Ushaw for the help and guidance throughout the course of my studies, and for providing me with the chance to share my knowledge and interest in video game technologies through teaching undergraduate and postgraduate students.

Sections of chapter 4 relating to patient trials of a software system designed and created by the author were done in conjunction with members of Prof. Janet Eyre's research team at the Institute of Neuroscience at Newcastle University, namely Sara Graziadio and Kholood Shalabi, who conducted most of the work involving face-to-face time with trial participants.

Abstract

There is an increasing body of work showing that video games can be used for more than just entertainment, but can also facilitate positive physical and mental changes. For people suffering debilitating side-effects from illnesses such as stroke, there is need to deliver and monitor effective rehabilitative physical therapies; video game technologies could potentially deliver an effective alternative to traditional rehabilitative physical therapy, and alleviate the need for direct therapist oversight.

Most existing research into video game therapies has focussed on the use of off-the-shelf games to augment a patient's ongoing therapy. There has currently been little progress into how best to design bespoke software capable of integrating with traditional therapy, or how to replicate common therapies and medical measurements in software.

This thesis investigates the ability for video games to be applied to stroke rehabilitation, using modern gaming peripherals for input. The work presents a quantitative measurement of motion detection quality afforded by such hardware. An extendible game development framework capable of high quality movement data output is also presented, affording detailed analysis of player responsiveness to a video game delivered therapy for acute stroke. Finally, a system by which therapists can interactively create complex physical movements for their patients to replicate in a video game environment is detailed, enabling bespoke therapies to be developed, and providing the means by which rehabilitative games for stroke can provide an assessment of patient ability similar to that afforded by traditional therapies.

Table Of Contents

List Of Figures

List Of Tables

Chapter 1

Introduction

1.1 Introduction

The advancement of affordable computer hardware has led to interactive media becoming pervasive in the modern world; from accessing online banking on a web-enabled television, utilising social media on a smart phone, to playing a computer game in a wide variety of genres, it is now common for people of all ages and backgrounds to in some way interact with a computer regularly. The increasing integration of computing into daily life has resulted in an ever growing body of work into how to effectively utilise interactive software for instruction, education, cognitive behavioural therapy, and physical and psychological therapy.

Since the introduction of affordable video gaming hardware such as the *Magnavox Odyssey* and *Atari 2600* in the 1970s, the video game industry has grown immensely; with a world wide revenue of $93 billion in 2013 [Gar13], it now rivals that of movies. Games are increasingly used for more than just entertainment - games such as *MineCraft* (minecraft.net; *Mojang*) are used in schools to teach concepts such as teamwork, geography, and even electronic circuits using the interactions between the in-game blocks that form the world.

1.2 Gamification

The popularity of games has led to an increase in the usage of *gamification*, in which the completion of tasks is augmented with game-like mechanics, to afford some degree of reward to what might otherwise be a tedious or undesirable task. Gamification concepts such as rewarding 'players' with 'experience points' or 'reward badges' have become common enticements to generate content on websites such as Wikipedia (www.wikipedia.org; *Wikimedia Foundation, Inc.*) and GitHub (www.github.com; *GitHub, Inc*). By utilising such reward devices, participants are provided with a sense of achievement, and measures by which to compare their

relative success against others, allowing an element of competition and fun to be injected into tasks.

1.3 Rehabilitative Gaming

The ability to use computer software to simulate situations and provide learning experiences, and to enhance player participation via gamification techniques has become a research topic in recent years as a tool for physical rehabilitation. Many common physical ailments, from frozen shoulder [HLY+14] and torn ligaments [Kvi04], to more serious conditions such as those stemming from the after effects of stroke [TSONB99], can be alleviated so some degree via physical rehabilitation; generally taking the form of intense and repetitive exercises designed to progressively strengthen a muscle [NHSb], or reinforce neurological changes pertaining to muscle motor control, known as *neuroplasticity* [KJ08]. These rehabilitation routines are generally quite tedious to complete [Las03], painful, and disheartening when a requested exercises cannot yet be completed successfully. Due to this, it is often the case that despite the benefits of long term compliance with a rehabilitative routine, patients do not comply fully with the physical therapy, or even withdraw entirely [JMMG10].

There has been work into bringing gamification to physical rehabilitation, by way of video games that present to the player high scores, unlockable extras, and achievements. This has largely been facilitated by the commercialisation of devices capable of tracking a person's movement in some manner, allowing determination of compliance with physical actions, rewarding the player upon success.

1.4 Motivation

Despite the increase in availability of affordable motion controllers, there is currently little in the way of commercial software utilising them that is specifically designed to help those with debilitating physical and neurological conditions, nor a standardised method of measuring the efficacy of such a game. Instead, most games advertised around some concept of promoting 'healthy' movements games are based around 'aspirational' principles such as balancing better or dancing for longer, utilising traditional try/reward gameplay mechanics, rather than being based upon engagement with true medically accurate feedback. Without such measures, it cannot truly be determined whether a game designed for rehabilitation is as effective at its purported goal as possible. It may also be the case that for persons with particularly debilitating conditions, that there is no game designed for public consumption

that can truly meet their requirements, requiring bespoke software. Therefore, there is a need for greater medical expertise to be involved in the creation of potential video games with rehabilitative benefit, to bridge the gap between games designed purely for entertainment, and for those designed around the promotion of health.

1.5 Contributions

The body of work described in this thesis investigates how best to create video games with a rehabilitative benefit, with particular focus on the physical therapies employed for those suffering from the effects of stroke. Such video games require some means of accurately recording user input, processing that input to determine the movements being attempted, and some way of engaging with the user in a manner which promotes successful engagement with the therapy.

To meet this goal, work was undertaken in three related areas, resulting in the following scientific and engineering contributions:

- A quantitative evaluation of commercially available motion and force sensing devices is presented and discussed, to provide insight into the applicability of such devices for rehabilitative purposes. Physical rehabilitation routines are likely to contain motions designed to test not only a subject's ability to move their limbs into the correct position, but also a specified orientation. For this reason, metrics for determining the suitability of commercial gaming devices to afford this information to rehabilitative software are presented, and a number of commercially available devices are benchmarked, with the influence of the local environment also discussed. For therapies requiring the detection of force rather than motion, a number of devices capable of measuring force are benchmarked for their accuracy and sensitivity.

- A configurable game framework, capable of utilising commercial motion and force sensing devices to determine high quality spatio-temporal data is introduced. Using the framework, a game suitable for use as a research platform into how best to create engaging routines for the physical rehabilitation of acute hemiplegic stroke is presented. The high quality output and configurable nature of the framework are utilised to show how changes to the instructional information presented by rehabilitative software can have a measurable and significant impact on its efficacy as a tool for providing long-term motor control improvements in those suffering from the effects of stroke.

- A general-purpose method by which rehabilitative routines based on replication of movements can be created, modified, and deployed to patients undertaking rehabilitation therapy via gaming. Utilising commercial gaming mo-

tion controllers, movements can be specified and edited by occupational therapists, before being integrated into the rehabilitative game. Using the motion detection engine outlined in this thesis, a measurement of a patient's ability to replicate and complete a previously recorded movement can be obtained, informing both the rehabilitative game of success levels, and occupational therapists involved in a patient's care as to their level of motion impairment.

These contributions provide the groundwork for the design and implementation of video games designed with a rehabilitative benefit in mind. By enabling devices to be quantitatively tested and integrated into third party software, and allowing complex, bespoke, motions to be recorded and tested for successful replication, games designed for therapeutic outcomes can be more effectively created; the work further provides insight into best-practises when designing and implementing rehabilitation-friendly gameplay.

1.6 Structure Of Thesis

The thesis is organised into 6 chapters. Further to this introductory chapter, Chapter 2 consists of background and related work in the field of rehabilitative gaming. The following three chapters contain the work performed for each of the contributions outlined in Section 1.5. As each chapter presents a different aspect of the overall work performed, each contains all discussion and evaluation pertaining to the work described within it, rather than all such material being collated into a dedicated results chapter.

Chapter 3 consists of a description of the metrics and testing procedures used to provide qualitative measurements of the quality of data output from motion and force-sensing hardware devices. A number of such devices are tested, and evaluated for their suitability in the context of a video game based rehabilitation.

Chapter 4 pertains to the work undertaken to demonstrate the sensitivity of rehabilitation oriented gaming applications to changes in the presentation of instructional data. A configurable game development framework is described, as is a game built using it to examine the use of force-detecting devices as input devices for uses post acute stroke. Trials performed using the game to establish a link between motor-task descriptions and motor learning ability are also described.

Chapter 5 focuses on the creation of a general purpose gesture creation and detection system, designed for use by occupational therapists focussing on a patient's rehabilitative regime, and to provide useful feedback as to the user's ability to successfully replicate the gesture. The methods of interfacing with motion devices are

outlined, as is the mechanism by which the system detects how well a user performed when attempting to replicate previously recorded gestures. A description of the integration of the system in conjunction with a number of third parties is presented, to demonstrate the system's ability to be used within the games industry to ease the creation of professional quality rehabilitative games.

The thesis is then brought to a close with a the conclusions presented in Chapter 6, which additionally includes a summary of the work undertaken, and a discussion on possible avenues for future work.

1.7 Publications

Over the course of this research project, the author has participated in a number of publications in related fields, outlined below:

Gary Ushaw, Richard Davison, Janet Eyre, Graham Morgan: Adopting Best Practices from the Games Industry in Development of Serious Games for Health. *5th International Conference on Digital Health. 2015*

Gary Ushaw, Richard Davison, Graham Morgan, Janet Eyre: Benchmarking Motion Sensing Devices for Rehabilitative Gaming. *Proceedings of the 30th Annual ACM Symposium on Applied Computing. 2015*

Gary Ushaw, Richard Davison, Graham Morgan: Commodity Video Game Technology in Teletherapy. *Handbook of Digital Games and Entertainment Technologies, Springer. 2017*

Richard Davison, Sara Graziadio, Kholood Shalabi, Gary Ushaw, Graham Morgan, Janet Eyre: Early Response Markers from Video Games for Rehabilitation Strategies. *ACM SIGAPP Applied Computing Review archive Volume 14 Issue 3, September 2014, Pages 36-43*

Sara Graziadio, Richard Davison, Kholood Shalabi, Kalvin M A Sahota, Gary Ushaw, Graham Morgan, Janet A Eyre: Bespoke Video Games to Provide Early Response Markers to Identify the Optimal Strategies for Maximizing Rehabilitation. *Proceedings of the 29th Annual ACM Symposium on Applied Computing. 2013*

The following was also published during the course of undertaking the work within this thesis:

Ben Kenwright, Richard Davison, Graham Morgan: Dynamic Balancing and Walking for Real-Time 3D Characters. *Proceedings of the 4th international conference on Motion in Games. 2011*.

Ben Kenwright, Richard Davison, Graham Morgan: Real-Time Deformable Soft-Body Simulation using Distributed Mass-Spring Approximations. *The Third International Conference on Creative Content Technologies. 2011*.

Chapter 2

Background

2.1 Introduction

This chapter provides a background study into relevant academic literature pertaining to the usage of video game technologies for the purpose of training and rehabilitation, with 'video game technologies' used as a term to encompass both software and hardware traditionally used for the purpose of interactive entertainment. Gamification and computerised simulations of environments has become a large research area of interdisciplinary research, therefore this chapter focusses on usage only within a medical context. This chapter therefore provides the context within which to justify and frame the work within in the following chapters, in which the author's own contributions are detailed.

2.2 Health Gaming

The literature indicates that there has been attempts investigated video game technology for their amenability at augmenting therapies for a wide range of ailments, including cerebral palsy, stroke, age-related balance problems, and other, non-physical problems such as memory loss. While the work within this thesis is designed around the needs of stroke, there is significant overlap in the types of disability these ailments can bring, making investigation into a range of health problems a worthwhile endeavour.

There is a pressing need for greater availability of affordable and effective rehabilitation mechanisms for those suffering neurological and physical impairments; in regards to stroke, it has become a major health issue world wide, with over 795,000 people suffering a stroke in the US in annually, and someone dying of stroke on average once every 4 minutes [GMR+13]. The detrimental side effects of stroke are many, including problems with motor control and balance [DSR+03], along with loss of vision, speech, memory, and limb movement [NHSc]. Such limb

dysfunction can take the form of upper or lower limb weakness, or in weakness or paralysis of one side of the body, conditions known as *hemiparesis* and *hemiplegia*, respectively [Tru]. It has been shown that improvements can be made to the range of movement, accuracy, and speed at which the affected limb can be improved via rehabilitation regimes [DS97] [DRW+98], often involving repeated intensive physical effort, such as repetitive stretching and gripping. The intensity of these rehabilitation procedures can affect the overall improvement in condition post-stroke [KWK+97].

Such regimes must be overseen by qualified therapists, a limited and expensive resource, who often have many patients to see, leading to limited 'face to face' time in which to effectively asses a patient's progress [CTR+15]. Additionally, despite the potential benefits of such training, the physical difficulty and possible slow progress can result in patients withdrawing from rehabilitative regimes early, greatly reducing their impact potential, with some reports stating less than a third of patients prescribed self-directed home rehabilitation actually performing the exercises [IAH11], although other works describe greater compliance in home-based therapies that still have active therapist interaction than those which require travel [DSB+11]. Despite the the health benefits of traditional physical therapy methods, it has also been reported that only 31% of post-stroke patients exercised regularly [SRM06]. Therefore, augmenting the rehabilitation process with gamification concepts such as positive reinforcement and achievements could lead to a person's greater engagement with their wellbeing, and potentially improving their quality of life.

For these reasons, it would therefore be beneficial to seek out methods in which an effective form of physical rehabilitation can be undertaken with little supervision from therapists, and in such a manner that continuing with the therapy is both encouraged and desirable. To meet this goal, it has become increasingly popular for video games and associated technologies to be brought into rehabilitation strategies. By combining a rehabilitation regime with video game technology, it is possible to create a standardised, entirely repeatable environment with tightly controlled task-based gameplay and patient feedback, that can also be engaged with in an enjoyable way that increases motivation to complete therapy [BDN+07]. To maintain patient engagement with the software, a level of personalisation is desirable, as tasks relevant to the patient's unique disability are most likely to assist in their recovery, with task difficulty set such that there is a good balance between challenge and achievement, encouraging engagement long-term [BMC+09]. A correctly developed rehabilitative game should also allow for an assessment of patient progress using standard medical metrics, much in the same was a therapist will as-

sess a patient's progress using validated metrics such as the popular Fugl-Meyer Assessment[FMJL$^+$75] used for assessment of post-stroke ability.

While much popular media focus on gaming and health has been on negative aspects such as violence and addiction, research into the potential mental benefits of actively engaging in games has been conducted since gaming technology began making traction in the home, with early studies looking into the benefits of playing games on slowing down the effects of ageing on mental health [Wei83]. Since then, the use of gaming as a tool in learning and medical improvement has become more commonplace. In the specific context of stroke rehabilitation research, video games are often combined with bespoke hardware, designed specifically for detecting the movements of the player's limbs, to determine engagement with some desired movement.

The concept of using video game systems to present interactive entertainment designed to in some way improve a person's health or wellbeing has become increasingly popular as such systems gain in popularity; in particular since the focus on movement tracking hardware prevalent in the 'seventh generation' of console hardware ushered in by the release of Microsoft's Xbox 360 console (www.xbox.com; *Microsoft, Inc.*) in late 2005. As research into beneficial entertainment products has increased, a variety of terms have been used within the literature to describe the concept, including *Exergaming* [SHM07], *Serious Games* [Ann10], *telerehabilitation* [RDPPC$^+$12], *wii-habilitation* [DLLP09], as well as the more broader term of *gamification* [JTM$^+$13].

Benefits to utilising gaming technology in health research include the ease in repeatability of experiments due to the deterministic execution of the game, the affordability of hardware, and improvements in patient compliance with their given tasks[RCB$^+$08]. The addictive nature of video games, presented as something to have fun with rather than endure, appears to lead to this increase in compliance. Early works demonstrating this effect include Warburton et al. [WBH$^+$07], where in obesity prevention trials in overweight young males, attendance of video-game based interventions was approximately 30% higher than for more traditional methods, in which attendance gradually lowered as participants lost interest. The video-game intervention took the form of a Sony PlayStation 2 (www.playstation.com; *Sony Computer Entertainment LLC.*) console attached to an exercise bike, with an external apparatus used to transform cycling speed to an analogue trigger amount, and the handlebar movement to analogue joystick movement; together, this allowed a variety of games to be played such that the player's 'real life' physical performance influenced their in-game avatar's position and velocity. The work goes on

to show that this system was at least as good as the traditional intervention in measured parameters including cardiorespiratory response and musculoskeletal fitness; partially due to increased attendance stemming from game enjoyment.

2.3 Demographics

Of the 971,000 hospitalisations for stroke in the U.S in 2009, the mean age of the patients was 71, with fully two thirds aged over 65 [HLD12]. For this reason, any attempts at integrating video game technology into stroke therapies must take into consideration that the patient may not be familiar with modern gaming devices, and gameplay genres.

The common stereotype of a 'typical' video game player is traditionally seen as that of a teenage male. However, market trends and recent research have shown that this is not necessarily the case. A report by the Entertainment Software Association [Ent14] states that the average game player is actually 31 years of age, and that women make up nearly 50% of the gaming population. The report goes on to state that, contrary to popular belief, there are almost twice as many women over the age of 18 than males under 18 that regularly play games. Further statistics from the report include that games defined as *casual* or *social* make up the largest percentage of played genres within the mobile and online video game markets - terms used to describe simple puzzle games with no deep story to be invested in such as *Candy Crush Saga* (candycrushsaga.com; *King.com Ltd.*), or which have a collaborative element via social media, such as the popular game *Farmville* (zynga.com/games/farmville; *Zynga Inc*). A 2010 report by *Popcap Games*, developer of the popular *Bejeweled* (bejeweled.com; *Electronic Arts Inc.*) casual game series, states that only 6% of social gamers are under 21, a smaller percentage than the 16% of those over 60 [Gro10]. Taken together, these reports indicate that older people as a whole are not against playing video games, but that the types of games played by young and old people are likely to differ, and as such gaming based rehabilitation may need to be packaged into different types of gameplay in order to ensure that a video game based medical intervention is adhered to.

Beyond this, there is little further research into the percentages of people over 65 playing video games. The research that has been conducted indicates that people within this age group are likely to watch more television than younger people [DSTJ13], but how this relates to video game usage is unknown. One study indicates that middle aged and older patients accounted for more than 50% of physiotherapist workload [HCDC04]; it is therefore important to take into consideration the likelihood that they may not have even played a video game, and may therefore

be unfamiliar with common gaming features such as multiple lives, high scores, or how the D-pad on a joypad maps to on-screen movements. The relevant literature presented within this chapter does indicate that elderly people are not against attempting to play games with appropriate support, and a recurring motif is of enjoyment and a desire to continue playing.

Video game based interventions are seen as a useful tool in tackling the problem of obesity in children [GPHS09], a section of society whose rising obesity levels has been linked to increased time spent in front of a television [DG85][And98]. The types of gameplay employed in obesity interventions is important, with differences noted in energy expenditure between healthy and obese children [ORH14], indicating that careful considerations must be made in the application of gaming technology for health benefits.

2.4 Measures of Rehabilitation Progress

In order to determine the efficacy of a particular physical rehabilitation therapy, it is common for those partaking of the therapy to be tested with a number of common metrics, depending on the exact form of disability they are suffering from. These metrics are usually standardised, and extensively tested to ascertain their effectiveness at determining the exact nature and severity of a person's disability. There are a wide variety of different performance metrics used within the metrics, designed to measure a person's balance, core muscle strength, arm and hand dexterity, and so on. Generally, these standardised metrics are based around either how long it takes the patient to complete a specific task, or a therapist's subjective determination as to a patient's ability at a specific task.

For a measurement to be usable comparative metric, it must be be proven to be as consistent as possible - as measurements are usually recorded by a therapist, there is potential for subjectivity in the result. This consistency is usually determined by its *reliability* and *reproducibility*. Ideally, a measurement should have high *inter*rater reliability (that is, multiple therapists should determine highly similar results for a measurement), and high *intra*rater reliability (the same therapist should determine highly similar scores from multiple tests of the same test subject). This reliability is usually measured as the correlation between datasets, either across multiple sessions, or across multiple raters; where possible, this thesis includes these correlation measurements when cited in relevant literature.

For measurements that are determined via a combination of individual tests, the concept of *internal consistency* becomes important. That is, all of the individual

tests should be influenced by the same testing criteria - if across 10 tests there is one with a consistently large difference in score than the others, it is likely that it is a poor fit for the overall phenomena being measured.

As part of a patient intervention, it would be invaluable for any piece of rehabilitative gaming software to be capable of providing metrics that are reliable, reproducible, and consistent. High reliability should be inherent to the software, barring any variations from random gameplay elements, or non-determinism of input polling and thread scheduling, while consistency will come as part of the design and testing process of the software development. A well implemented medical metric within the software would theoretically afford much higher fidelity in measurement accuracy and possibility of automated feedback so as to ascertain why a test has resulted in a particular score.

In order to gain a greater understanding of the types of data typically used within patient therapy, an overview of metrics commonly found within the relevant literature is now provided.

2.4.1 Balance Measurements

2.4.1.1 Berg Balance Scale

When determining the level of ability in older people, the *Berg Balance Scale* [BWDWG89] has become a standard method of providing a quantitative measurement of patient balance. The Berg scale is determined by the result of scoring 14 requested actions from 0-4, resulting in a final maximum score of 56; a score of 40 or higher is considered an indicator of the participant being capable of full independence. The scoring actions include sitting unsupported, standing with eyes closed, rotating 360 degrees, and the ability to reach forward in a standing position. Positive aspects of the scale are its internal consistency (that is, the results of the 14 actions are tightly correlated), and the reliability of similar results across different testers and test sessions.

2.4.1.2 Timed Up & Go

Another popular metric within the literature for determining the overall mobility of an elderly is the *Timed Up & Go* test [PR91]. The test itself is simple - the patient must begin sitting, and then stand, walk forward for 3 metres, before turning around and sitting back, with the recorded metric being how long this process takes. While simple, the task requires coordination, balance, and spatial awareness to complete in a timely manner. Results of greater than 30 seconds are seen as indicators of a patient being at risk of fall. The simplicity and lack of subjective parameters gives

the test a high level of repeatability and reliability.

2.4.1.3 Functional and Lateral Reach

Further metrics cited within the literature when used for verification video game based interventions include *Functional Reach* test [DWCS90] and *Lateral Reach* test [BBG99], used to determine the maximum stable reach of a patient in the anterior and lateral plane, respectively. In both cases, this is performed by standing still, and attempting to reach specified targets. A force sensing plate is also used, to determine the centre of balance while performing the tests. As with other noted measures, repeatability and reliability [BBD+03] are drivers of their popularity, as well as the simplicity of their execution. Such reaching metrics could be replicated using modern motion tracking gaming hardware, if the fidelity of their output is of sufficient quality.

2.4.2 Motor Control

2.4.2.1 9 Hole Peg Test

To test fine manual dexterity, the *Nine Hole Peg Test* has become a standard method, used by therapists worldwide. The test has been shown to have capable of high reliability [ECE+11]. The test itself is simple - a board with 9 holes laid out in a 3x3 pattern (as seen in Figure 2.1) is placed in front of the person to be assessed, along with 9 pegs. The test metric is then the number of seconds taken to insert all of the pegs into the holes, and then remove them again. The test is to be performed with only one hand, making it suitable for dexterity testing in subjects with hemiplegic disability. The test has been tested extensively with children [PBT+05], and adults [MWKV85], to determine norms for the peg test at a number of age ranges.

There has been work towards creating a virtual replication of the peg board test [FLG11] utilising a *PHANTOM Omni* (www.dentsable.com/haptic-phantom-omni.htm; *Geomagic Inc.*) haptic device capable of positional sensing and force feedback. It was found that people suffering from stroke could operate the device and complete the peg tasks. This raises the possibility that such a test could be added to a rehabilitative game as a validation stage, when coupled with a suitable input device.

2.4.2.2 ABILHAND

The *ABILHAND* [PTT98] measure is a common clinical tool for upper limb ability found throughout the literature pertaining to therapies for cerebral palsy. It consists of a number of individual tasks (commonly 23 [PTA+01]) delivered via questionnaire, to be scored between 0 and 3 based on the subject's opinion of their ability at performing the stated task. Tasks include opening an envelope, squeezing

Figure 2.1: An example of a commercially available 9 peg board, used as a measure of motor control; image via Amazon.com

toothpaste, and washing both hands; not all tasks are assumed to be of the same difficulty. Results are weighted against difficulty and summed to provide a final raw score, which is then converted to a final ABILHAND measurement value. The original author of the ABILHAND performed validation of the test for adults with hemiplegic stroke [PTA+01]. The test has also been used in an adapted form for the validation of manual ability in children, and validated for measurement for children with cerebral palsy [APRT04].

As the measurement is a basic question/response metric, it is easy to see how it could be computerised and used within rehabilitative software. It is not particularly suited for implementing as a gameplay component, due to its limited interactivity, but could be utilised as a periodic pre-game questionnaire, with trends over time analysed.

2.4.2.3 Melbourne Assessment

Superficially similar to the ABILHAND-Kids metric is the *Melbourne Assessment* [RJR99], designed for determination of upper limb ability in children between 5 and 15 years of age with cerebral palsy. Unlike ABILHAND however, the Melbourne Assessment requires the test participant to actually attempt the tasks required, with the entire test procedure recorded for later analysis for score determination. Test subjects are required to complete 16 tasks, with each tasks graded on a number of metrics, such as range of movement, accuracy, and fluency of movement. Tasks include reaching forwards and sideways, crayon manipulation, and demonstration of hand to hand transfer of items. The result is a percentage based on the summation of all scored metrics.

The assessment has been determined to be reliable [RCCR01], with moderate to high intrarater reliability, good interrater reliability, and high consistency of data

Figure 2.2: An example of the Wolf Motor Assessment of motor function: Timed task 18, turning a key

(0.97). As the assessment requires actual props for manipulation during the tasks, the test would be difficult to successfully replicate without also providing the props. The grading metrics however, including the range and fluency of movement, could be recorded using modern motion tracking devices, so tests derived from this assessment would be possible to be integrated into a virtual rehabilitation program.

2.4.2.4 Wolf Motor Function

The *Wolf Motor Function Assessment* [WLBJ89] is used to assess upper extremity function in patients suffering from disorders that limit motor function, including stroke. As with other standardised medical tests, there is a strict set of steps, with appropriate scripted instructions, that the therapist must go through with the patient, in order to correctly produce an assessment score. The assessment itself is comprised of 21 individual stages - 18 timed activities, with results measured in seconds, and 3 force tasks measured in pounds. Timed activities are grouped by target joints, and include movements such as shoulder utilisation to lift the forearm onto a box, elbow extension to pick up an object placed in front of the subject, and time taken to turn a key, as shown in Figure 2.2. Force measures include lifting the tested hand onto a box while weights are attached, and grip strength measured using a dynamometer. Later evaluation of the test [WCE+01] when used with post-stroke individuals noted that the WMFA had very high interrater reliability (0.95 to 0.99), and its individual stages shown to have good internal consistency.

As with the 9 Hole Peg Test and Melbourne Assessment, the WMFA utilises props for a number of its stages, such as lifting of a pencil and paperclip, and stacking of checkers. While the physical movements required for these tasks could be recorded using motion tracking hardware such as the Kinect, or a finger tracking device such as the Motion LEAP for the finer tasks, the props would have to be provided in

Figure 2.3: The Wii Balance Board device on Step Risers, a combination that could replicate standard motor function assessments such as the Timed Up and Down Stairs Test

order to be standardised, and some of the larger props (task 20 involves lifting a basket) would affect the accuracy of camera-based motion detection mechanisms, making a full replication of the WMFA using video game technology infeasible. It would be possible to record a more limited number of the tasks involving simple limb extensions and rotations to provide a standardised measure of player progress, however.

2.4.2.5 Timed Up & Down Stairs

A more recent test of overall motor function, the *Timed Up and Down Stairs Test* [ZMW04], has been used in literature pertaining to cerebral palsy therapies. The test is simple, with the participant being requested to climb a 14 step flight of stairs, before turning around and descending back to the bottom. The metric is the number of seconds taken to complete the task, with lower values assumed to be greater overall control of motor function. The lack of subjectivity in the metric affords excellent intra- and interrater reliability [ZMW04]. The nature of the required prop (a 14 step flight of stairs) makes exact replication of the TUDS test using video game technologies impractical. It is possible however to create a reasonable facsimile of the test using the *Wii Balance Board*, a force sensing device that will be more fully described later in the thesis. Aerobic stepping games have been released (www.zoozen.com/stepup/; *Zoozen*), designed to be played with the Wii Balance Board mounted on risers to increase step height (example shown in Figure 2.3). Counting steps via the balance sensors in the Wii Balance Board would afford a similar metric to the TUDS test, perhaps with step quality measured utilising more advanced sensing hardware, such as the skeletal tracking system afforded by the Microsoft Kinect device, also discussed in more detail later in the thesis.

2.4.3 Measurements for Assessment of Stroke

2.4.3.1 CAHAI Assessment

The *Chedoke Arm and Hand Activity Inventory* [BGS$^+$04] is a common metric for upper limb motor control for post-stroke patients, and designed to assess the ability

to perform 'real world' tasks, indicating the level of independence of the test taker. A CAHAI assessment is made up of a number of sub tests, each of which is graded by the testing therapist based on an ordinal scale between 1 and 7, with 7 indicating 'complete independence', down to 1, representing 'total assistance' required to perform the given test. The final CAHAI score is then the simple summation of these individual grades, and thus will usually range between 13 and 91. As with the Wolf Motor Function test, a number of everyday items such as a ruler and coffee jar [BSM+] are used props during the assessment, in keeping with the concept of assessing motor function for 'real world' tasks. Due to the number movements tested, time spent assembling props and describing tests, an assessment session may take over an hour to complete, tying up a therapist for the duration.

The CAHAI has been extensively used throughout stroke rehabilitation literature, and used as a comparison metric in trials involving video game delivered therapies. The original creators of the assessment performed validation of the metric [BSL+05], finding that its sub-tests had high internal consistency (0.98), and that the test demonstrated a high interrater reliability, despite the subjective nature of the sub-test result scale. An investigation by Rowland et al. [RGT+11] found that polled occupational therapists thought a reduced, 9 item CAHAI was a useful metric for mild to moderate upper limb function deficit, but less suitable for those with severe problems. Therapists have reportedly found CAHAI hard to assess correctly, and patients have stated that the nature of instructions makes it hard to know how to complete moves when the nature of their disability precluded completion as per instructions given[GTD10]. This raises an interesting point in relation to instructions - a video game based rehabilitation may need to detect when a player is not performing a movement correctly and correct them via visual or aural feedback, as otherwise, without a therapist to correct them, they may continue to perform incorrectly, potentially lessening the impact of the therapy, or even physical injury.

2.4.3.2 Fugl-Meyer Assessment

Within the literature pertaining to video game rehabilitation for those suffering from hemiplegic stroke, the most commonly used metric is the *Fugl-Meyer Assessment of Motor Recovery after Stroke* [FMJL+75]. Similar to the CAHAI score, the Fugl-Meyer assessment is split into five steps, each consisting of a number of stages, with the assessee's success at each stage graded either 0,1, or 2; the summation of these grades for each side determines the final assessment score. While the CAHAI was designed to grade the upper limb function only, the Fugl-Meyer is divided between the upper extremity (the forearm, elbow, and shoulder), and lower extremity (the hip, knee, and ankle), with individual tests for each segment of the limbs. For example, an upper extremity step requires the assessee to rotate their wrist such that

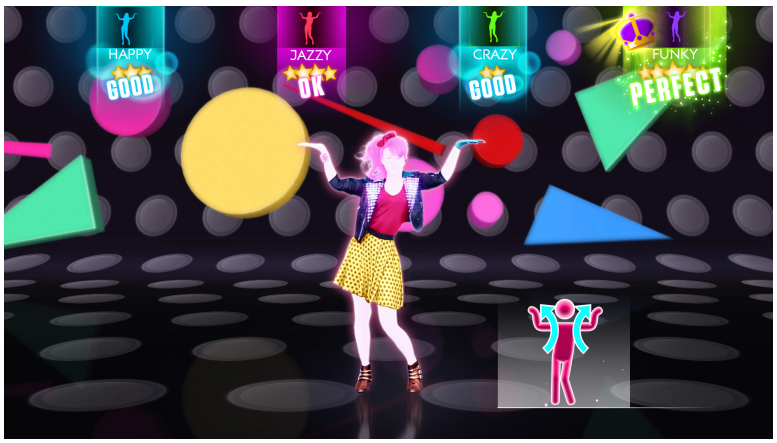


Figure 2.4: The on-screen gesture graphics in *Just Dance* allow the player to easily see what to do to succeed in the game

their hand faces upwards, with a resulting score of 0 for no movement, 2 for fully successful, and 1 for partial success.

The Fugl-Meyer assessment has been extensively tested, and found to have a high level of inter- and intra-reliability [DPN83][SMS+93][PPvW+05][STC+11], and is noted as an effe ctive means of evaluating changes in motor impairment following stroke[GDB02].

Due to the complex nature of individual motor function requests in the assessment, it is permissible and recommended within the literature to explicitly inform the assessee how to complete each action, for example via miming the required action. The allowance of such instruction in medical assessment is an example of how video game related rehabilitation therapies can produce medically validated results; it is easy to imagine an on-screen graphical character demonstrating to the player via example on how to complete a task, an approach already taken in movement-based video games such as Ubisoft's popular *Just Dance* series (www.just-dance.ubisoft.com; *Ubisoft Entertainment Inc.*), as shown by the on-screen graphical instructions in Figure 2.4.

2.5 Video Game Therapies

2.5.1 Early VR

Early attempts at creating a rehabilitative environment akin to a video game include [SMT+03], in which a commercial 'green screen' Virtual Reality system named *IREX* (www.gesturetekhealth.com; *Gesturetek Health Inc.*) was utilised as part of two patient trials - one for the rehabilitation of the shoulder joint, and one for balance retraining in those with brain injury. Patients undertook a trial whereby they

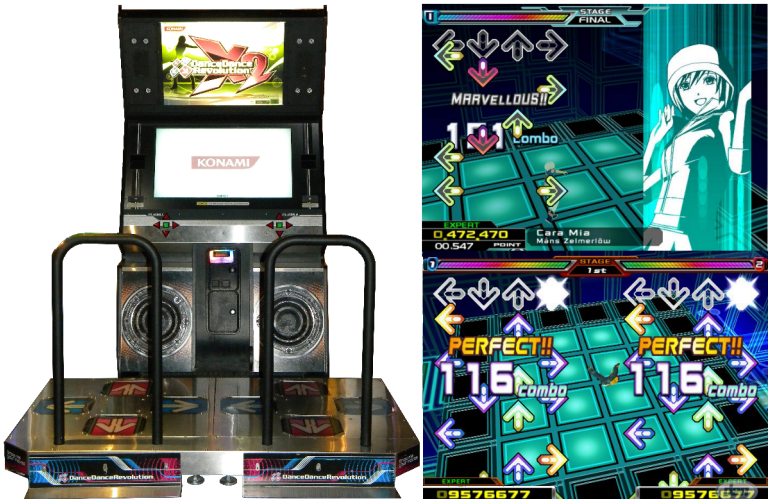


Figure 2.5: Dance Dance Revolution was an early target for health gaming research. Left: The original arcade machine with metal dance plate. Right: In-game screenshot showing the on-screen arrows that indicate the dance moves to perform

played a game in which they must act as a goalkeeper saving virtual footballs that are being 'kicked' at them in directions chosen by a therapist, either to match the range of motion in their shoulder, or their range of leaning and stepping, depending on the nature of the therapy required. Both methods led to improvements in the relevant medical measures. A later trial utilising the IREX system in the rehabilitation of a child with hemiparetic cerebral palsy [YJK+05] resulted in neuroplastic changes and limb coordination improvements by playing a suite of VR games for an hour a day for 4 weeks.

2.5.2 Dance Dance Revolution As A Rehabilitative Tool

One of the earliest commercial gaming products that was used to research the health benefits of video games was *Dance Dance Revolution* (us.konami.com/ddr; *Konami Digital Entertainment Inc.*) (commonly referred to as simply *DDR*), released as stand-alone arcade machine by Konami in 1998 (see Figure 2.5), and later as a home console release. In the game, the player stands atop a custom apparatus and is presented with a series of scrolling directional arrows on screen, corresponding to four arrows at their feet (placed at anterior, medial, lateral, and posterior positions), that the player must physically actuate in time to those on screen. Points are earned within the game for accuracy in both timeliness and correctness of arrow pressing, with the player being updated as to their progress using visual and aural clues (for example, they may be informed that they have beaten their previous record for uninterrupted success at pressing the arrows). This simple gameplay mechanic of having to physically move to win has proven to be incredibly popular, with over 6.5 million units sold in the home console release of the game [Gam], which utilised a plastic pad connected to the console rather than the metal chassis of the arcade release; this version of the game was further enhanced using concepts such as unlockable con-

tent and online leaderboards, enabling more competitive play and a sense of reward.

The popularity and unique input of the DDR game led to its use in a variety of research trials focussing on potential health benefits. Much of this work has centred around the possibility of using the game as a tool for tackling obesity, but the nature of the game mechanics make it suitable for use as a tool for patient balance and co-ordination research. Maloney et al. [MBK+08] is an example of research into how video games can be used to maintain a healthy lifestyle, in this case among children of approximately 8 years of age. In it, two trial groups of children are described, one of which was asked to play DDR for two hours a week for ten weeks, and another to forego playing the game for the same period. Data collected at week 0, week 10, and post trial at week 28 showed that the DDR group had a significant reduction in their sedentary screen time (a term equating to waking time spent looking at computer or television screens whilst engaging in little to no physical activity) over the control group, with high satisfaction from participants and anecdotal evidence of an overall increase in physical activity; together these support the notion that video games can work as an enabler for beneficial exercise.

A study undertaken by Murphy et al. [MCN+09] documents similar results. It shows the results of a 12 week trial of 35 children overweight children aged between 7 and 12, of which most were diagnosed as having *Metabolic Syndrome*, a particularly dangerous combination of hyperglycaemia, high blood pressure, and obesity that is a high risk indicator of diabetes [AZS05]. As in [MBK+08] Participants were separated into two groups, only one of which was asked to play the DDR game 5 days a week, and additionally wear pedometers. Pre and post trial patient metric readings showed that the DDR playing group had improved aerobic capacity, and gained less weight than the non-DDR group, with a trend towards a significant reduction in body mass index. Compliance was high, with three quarters of the DDR playing trial participants playing for at least 5 days a week across the 12 week trial; evidence that even those characterised by previous low exercise rates can successfully engage with a video game based intervention.

Sell et al. [SLT08] found in a small trial of mostly male college age students that regular players of DDR met or exceeded recommendations for healthy physical activity by playing the game, although the authors note that the narrow scope of the study places limitations on its reach. A DDR-style dance game was used in a children's trial detailed in Maddison et al. [MMJ+07] in which 21 children between the ages of 10 and 14 played a variety of video games, and found DDR to be equivalent to physical activity of moderate intensity when compared by metabolic equivalent, which could lead to weight loss if played regularly; however this particular trial had

no data on trial compliance to suggest likelihood of this happening in the long-term. The equivalence of DDR to moderate exercise is again noted in the work of Graf et al. [GPHS09], in which a trial of 23 children found that DDR resulted in a 3.3 times increase in energy expenditure over resting.

A trial of similarly aged children was conducted in Lanningham-Foster et al. [LFJF+06], which found that playing DDR resulting in greater energy expenditure than sitting or walking, and that this was regardless of whether the child was considered lean or overweight, suggesting that obesity is not necessarily an inhibitor to engagement with a suitable intervention; while the study notes that the children enjoyed and engaged with the video games, the trial period was not long enough to determine whether this results in maintained interest over time.

In addition to work on obesity, the gameplay mechanics of DDR lent themselves well to medical research into alleviating balance problems stemming from illness and injury. Brumels et al. [BBC+08] compared the efficacy of balance training routines utilising the DDR game and Wii Fit Balance board (www.nintendo.com; *Nintendo Co., Ltd.*) against traditional balance training therapy methods. The DDR portion of the trial utilised the standard DDR dance mat, but augmented the usual DDR gameplay rules by having supervisory staff request that users play using only their 'non-balancing' foot (determined by asking trial participants which leg they would use to kick a ball, and which they would balance on), and to ignore any in-game moves that require usage of both feet. This gameplay was compared against similar balance tests using the Wii Balance Board, and an exercise cushion used in traditional balance exercises, each over a similar time period of 15 minutes per exercise session. By asking participants to undergo standard testing prior to, and at the conclusion of, the 5 week study period, it was found that both the DDR game and the Wii Balance Board elicited statistically significant gains in force plate and star excursion balance testing compared to the trial group using only traditional balance exercises. Additionally, it was found via post-trial questionnaire that trial participants felt that they engaged with, and gained more enjoyment from, the gaming trials.

Kloos et al. [KFK+13] utilised the console version of the DDR game as part of their research into intervention strategies for those suffering from the effects of Huntington's disease, a degenerative brain disorder that affects muscle coordination and balance. In the feasibility study described in the work, 25 individuals with a Huntington's diagnosis undertook a 12 week trial whereby 6 weeks of DDR, and 6 weeks of handheld video games. A manual procedure was undertaken whereby trial participants played the game under supervision for 45 minutes, twice a week throughout

the 6 week span allotted for the DDR-based intervention, with supervisors aiding in the selection of the gameplay elements such as song to dance to and the speed at which movements must be completed, with the speed parameter increased for those performing well by the in-game metrics. Before and after the trial period, each participant was measured against their ability to play the game, and in traditional medical measures such as gait, balance, and quality of life. Significant improvements were noted in some metrics, and compliance with the intervention was high - participants engaged with the routine, with the in-game feedback providing positive reinforcement that encouraged greater play and concentration, potentially facilitating motor learning.

Work outlined in Smith et al. [SSS+09] and Studenski et al. [SPH+10] utilises a bespoke game with a DDR-style dancemat to examine the usefulness of such games in maintaining balance and stepping control in the elderly. Training of stepping and balance can help older adults react quicker to slips, and reduce the risk of falling, making the investigation of video game technology suitable for the elderly a worthwhile research topic. Smith et al. used a modified DDR mat connected to a PC to record player metrics such as reaction speed and percentage of missed moves in a bespoke DDR-like game. Unlike 'true' DDR, the mat used in the trial consisted of four targets, placed to the left, right, and diagonal left and right of the standing area of the mat (see Figure 2.6). Gameplay was otherwise similar, with the aim of the bespoke game to correctly press the target as an on-screen graphical representation of it drifts up screen. In a series of studies involving participants of a mean age in the late 70s, it was shown that elderly people could participate in a DDR-like game, but only in a game setup designed with their needs in mind; whilst all of the trial participants could manage a target completion rate of 1 target every two seconds, only 1 in 10 could manage 1.5 steps per second, lower than a control group of young adults. While unsurprising, it is still evidence that games should take into consideration target audience when gameplay specifics are devised. Studenski et al. provides the results of a similar trial, set over a greater time period (24 sessions over three months, rather than the 3 single trial studies of Smith et al.), utilising a DDR-style dance mat, and custom game software.

2.5.3 Rehabilitation utilising the Wii Remote

The release of the Nintendo Wii (www.nintendo.com; *Nintendo Co., Ltd.*) console was of great interest to those researching medical uses for video game technology, as it provided an affordable gaming platform, with a unique, motion-based control method. Bundled with the console was the 'Wii Remote' device (shown in Figure 2.7), a hand-held controller that can detect its orientation and acceleration, and can determine a 3D position in relation to a provided IR light source, to be placed above

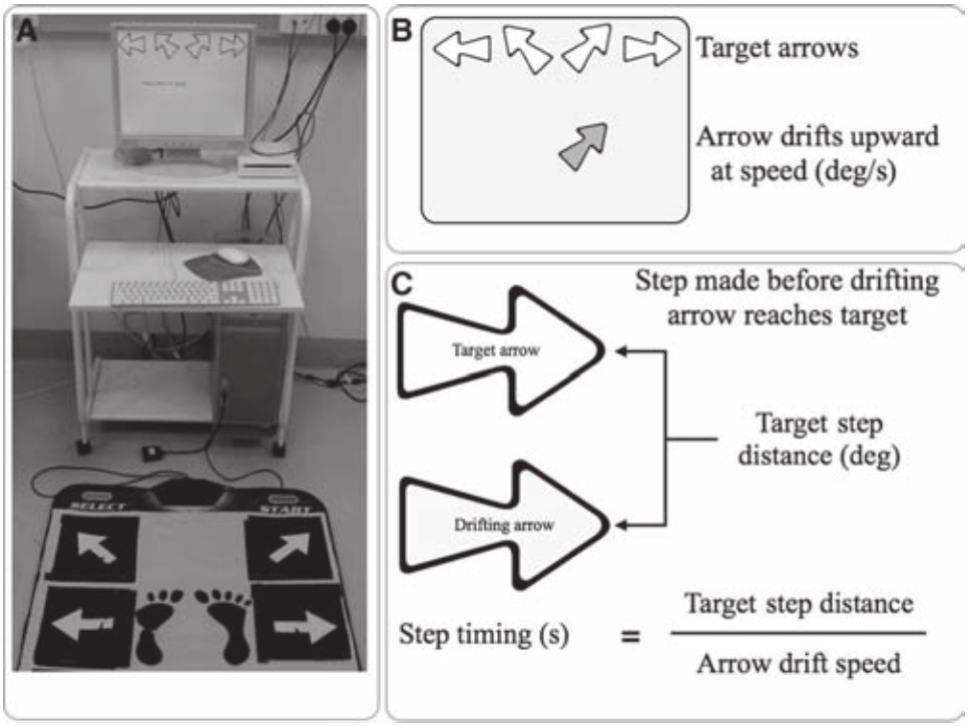


Figure 2.6: The modified dance mat layout employed in the work of Smith et al. has additional foot pads at diagonals in place of the straight forward and backward pads; this reduces the complexity of the movements while still maintaining the core DDR concepts of balance and coordination

Figure 2.7: Left: Nintendo Wii Remote. Right: A selection of acessories to attach to the Wii Remote to allow for greater replication of on-screen activities while holding the device

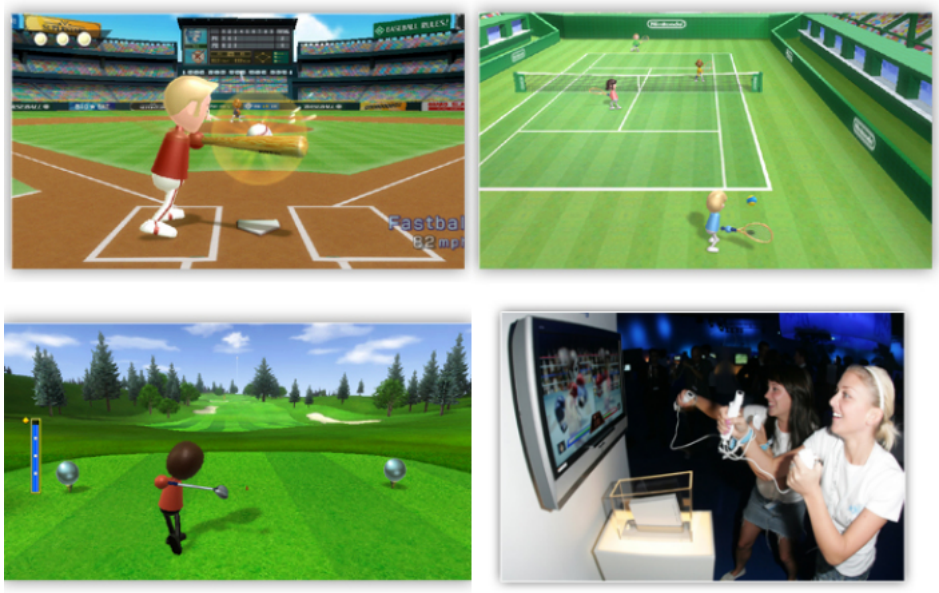


Figure 2.8: An example of the minigames that form the Wii Sports package. Clockwise from top left: baseball, tennis, boxing, and golf. Many of the subgames can be played with additional accessories attached to the Wii Remote

the user's television. A more detailed insight into the functionality of the 'Wiimote' device is provided in Chapter 3. One of the earliest Wii releases was Nintendo's own *Wii Sports* (as shown in Figure 2.8), a collection of minigames based on replicating 'real world' sports with the Wiimote - for example, in *Wii Golf*, the user must stand, holding the Wiimote as a golf club, and swing it to hit the on-screen ball. This replication of real world movements proved popular, with a number of plastic 'add on' devices for the Wiimote, designed to make the device look like the bat, club, or racket for the sport that the game replicates; Figure 2.7 shows a number of these accessories.

Perhaps the first usage of the Wii for rehabilitative purposes is in the work of Deutsch et al. [DBF+08], which describes using the *Wii Sports* software to complement the rehabilitation program of a child with cerebral palsy. Over the course of a 4 week intervention, the Wii Sports minigames were played in sessions lasting 60 to 90 minutes, which resulted in improvements in visual-perceptual processing, posture control, and functional mobility. The work notes that the participant exceeded standard times spent standing while playing, suggesting that engagement in a task is therapeutically beneficial. Further work by the same author [DBS+11] extended their research into the Wii towards post-stroke rehabilitation. The work describes an effort to categorise and assess the usefulness of the minigames that make up the Wii Sports and Wii Fit products. The games were assessed for their ability to target specific impairments (categorised as balance, coordination, strength, endurance, and upper extremity control), and the types and relevance of feedback presented to the player. The work presents one of the earliest discussions on the types and qual-

ity of feedback presented to the player, noting that the difference between reaching meaningful rehabilitative goals, and the tendency of games to be very loose in the needed accuracy of movement replication, means additional feedback may need to be delivered by therapists in-person. The work also makes note of the difference in game genre preferences between gender, and suggests that greater partnership between the fields of clinicians and game development companies would be desirable, something that this thesis in part attempts to address.

Shiner et al. [SBM14] describe a system by which, in conjunction with Wii-based rehabilitation using Wii Sports, patients suffering from hemiplegic stroke undertook a warmup routine whereby the plegic limb was connected to a mechanical armature that would physically move and rotate the limb in a mirror image to that of their non-plegic arm. Patients undergoing this extra warmup had additional functional improvements, and greater retention of such improvements over time than similar patients using the Wii rehabilitation regime alone, suggesting that there is scope for video game based rehabilitation to include neuroplasticity-enhancing techniques to maximise potential. Winkels et al. [WKT+13] undertook a trial similar to Shiner et al., again exploring the capability for the Nintendo Wii Sports game to be used in therapies for children with cerebral palsy. Within the trial (15 children with mean 8.9 years, opposed to 62 children with mean 9.4 years), it was again determined that no significant changes were found within the chosen metric of upper arm movement, although a slight significant improvement in two-handed daily activity, measured by the ABILHAND-Kids scale. As is common throughout the literature, reported enjoyment of the trial was high, suggesting that a game more specifically tailored to the requirements of the player could provide a significant benefit.

Further stroke related research into the Wii console was performed by Yong Joo et al. [YSX+10], which evaluated the feasibility of augmenting traditional stroke rehabilitation routines with gaming using the Nintendo Wii Remote. The evaluation took the form of a 16 participant trial over 6 sessions, in which they were asked to play Wiimote enabled games from the Wii Sports software for 30 minutes. Comparison between pre- and post-trial Fugl-Meyer assessments indicated a statistically significant improvement, but states that as there was no control group it cannot be guaranteed that the improvements were solely the product of teletherapy. The majority (81.3%) of patients described the video game trial as enjoyable, and expressed a desire for video games to be a regular part of their rehabilitation (87.5%); as with similar works, the mean participant age of 64.5 years indicates that age is not necessarily an indicator of adherence to video game related rehabilitation methods. The work notes that there were subjects incapable of correctly holding the Wiimote due to their disability, with the stated solution being to attach the Wiimote using the

strap or a bandage. Mass market gaming hardware is seldom designed for usability edge cases such as physical or mental disability, an issue which will remain an hurdle to a therapist-free deployment of a rehabilitation video game.

Mouawad et al. [MDMM11] details the usage of the Wii Sports game in aiding the rehabilitation of patients, in this case seven patients suffering the after effects of stroke. Unlike many other works within the literature, the authors utilise a control group of healthy people playing the same games as the test group, rather than the control group being those with similar disabilities, but not taking part in the video game based intervention. This allows some insight into whether the intervention truly has lasting effects on motor control, or whether it is primarily due to familiarity with the game. The trial protocol itself required participants to play Wii Sports across multiple sessions both at home and within a laboratory environment, for 14 days. Patients were encouraged to ignore on-screen indicators of progress, as the game design and rehabilitative outcomes were not particularly well matched. Patients were evaluated with the Wolf Motor Function Test, and also via the Fugl-Meyer test for motor control, and Berg Balance Scale for balance. Results of the trial included a reduction in mean Wolf Motor Function test time (from 3.2s to 2.8s per test), and an increase in mean Fugl-Meyer Assessment scores (from 42.3 to 47.3). This suggests that the types of movements and exercise encouraged by the Wii Sports package can result in real-world improvements. It cannot be determined precisely why the interventions for stroke result in a greater benefit than for therapies for cerebral palsy, but the difference in trial ages, and neurological differences in the conditions themselves could be factors.

The Wii Sports video game has been assessed in comparison to recreational therapy such as card games and board games, a traditionally recommended method of training dexterity in a recreational context. Saposnik et al. [STM+10] [SMB+10] performed a small 2 week intervention consisting of 22 participants, with 11 playing Wii Sports, and 11 playing leisure activities such as bingo, and the board game Jenga. Follow-up tests after 4 weeks of trial completion showed that the active group showed significant improvement in the Wolf Motor Function Test assessment over pre-trial testing, with a mean of 7 seconds improvement. It is not indicated within the work whether these improvements were maintained long-term. The repetitive, relatively intense, nature of Wii Sports over card and board games may determine the difference in groups.

Not all attempts at utilising the Wii Sports game have shown positive results. Chiu et al. [CAL14] describes a trial that utilised the Wii Sports game to aid in the rehabilitation of children with hemiplegic cerebral palsy. The trial of 62 children ranging

from 6-13 yeas consisted of 3 gaming sessions across 6 weeks of training using the Wii Sports game, in addition to any usual therapy undertaken. Their results indicated that there was no change between the trial group or control group in overall hand coordination, but with a slight difference in grip strength in the trial group over the control group. Hand function measurement via the peg board test also showed no difference. The work notes that while there was an improvement in trial participants in-game score, this evidently did not equate to any real-world improvement in hand function. While hard to draw conclusions, when compared against other works, the trial results would suggest that Wii Sports is perhaps poorly matched for cerebral palsy therapy, and that in-game scoring results can only be seen as a measure of familiarity with a game, rather than an absolute indicator of player capability.

Wii Sports has also been investigated for its usefulness as a therapy tool in those suffering from Parkinson's disease, a degenerative neurological disorder that leaves those afflicted with symptoms such as shaking, and difficulty walking. In Herz et al. [HMS$^+$13], a test trial of 20 patient's with Parkinson's disease is detailed, in which participants partook in 1 hour Wii Sports sessions 3 times a week. The trial also notes a slight, but significant improvement in depression symptoms, suggesting that even beyond physical rehabilitation, video games have the potential to improve quality of life.

As multiple Wiimotes can be connected to the same console or computer, there has been work into utilising the Wiimote as an inexpensive package for extracting accelerometer and gyroscopic data. In Alankus et al. [ALMK10a], work towards bespoke games for stroke rehabilitation is outlined. Multiple Wiimotes were attached to a patient's upper and forearm using velcro straps, allowing the approximate pose of the arm to be determined from the accelerometer data; additional devices attached to the body afforded more accurate determination of which muscle groups were used when performing a motion. A number of different games were created, designed to have varying motions and amount of cognitive challenge, as well as some degree of multiple player interactivity. Stated outcomes of the study included the need to calibrate devices to determine the amount of movement the user has in their limb, that games should encourage players to utilise and extend beyond their full range, and the requirement for games to have means of testing coordinated actions, due to their importance as quality of life measures. The work additionally notes that the game's response to user movement should be intuitive, i.e. that vertical arm movements should not be used to move a character left and right. The ability to play cooperatively and collaboratively with friends and therapists was also stated as important, as it afforded a sense of connection to the other player as an equal.

Figure 2.9: Utilising multiple Wii Remotes to determine limb pose by utilising accelerometer data from devices attached to the upper and lower arm, as used in Alankus et al. to control a number of bespoke video games for stroke rehabilitation

2.5.4 Rehabilitation utilising the Wii Balance Board

The global release by Nintendo of Wii Fit in 2008 was of great interest to the rehabilitative gaming research community, primarily due to the bundled peripheral, the *Wii Balance Board*. The battery powered Balance Board is shaped similarly to a set of bathroom scales (see Figure 2.10), and is designed to be stood on by the user, with the device detecting the user's weight and centre of balance using four internal pressure sensors. The accuracy of the device has been examined, and found by one source to be suitable for low-resolution applications, but inadequate as a replacement for laboratory-grade equipment [BTB14], and another as displaying high correlation between reported values and tested values when used to record weights [YFBC11]. The device is designed to connect wirelessly to a Nintendo Wii or Wii U, primarily for use with the Wii Fit product - a collection of smaller games with focus on body movement and balance; in one such game, the in-game avatar is a tightrope walker, with the aim of the game to maintain enough balance to successfully cross the rope. Beyond Wii Fit, unofficial support for the Balance Board was added to the linux kernel in 2012 [Tor], opening up potential for the device to be used in bespoke games beyond that of the Wii Fit programme.

Early works utilising the Wii Balance Board include Sugarman et al. [SWEBB09], a single patient feasibility study utilising Wii Fit in conjunction with standard physical therapy methods to aid in their post-stroke rehabilitation routine. The intervention took the form of four consecutive training sessions, each of 45 minutes in duration, with the Wii Balance board used with a safety railing, a feature utilised across many of the rehabilitation routines examined in the literature. Balance measures including the Berg Balance Scale and Timed Up and Go test were performed,

Figure 2.10: The Nintendo Wii Fit Balance Board, a Bluetooth compatible device capable of measuring weight and centre of balance using force sensors placed at its corners

with a 10 second decrease in TUG noted, a clinically relevant result. Despite an age of 86, the participant engaged with and enjoyed the Wii Fit game.

In Fung et al. [FHS+12] the Wii was investigated for its ability to aid in the rehabilitation of the elderly after knee replacement surgery. In a trial of fifty patients of a mean age of 68, all of which were outpatients following knee replacement, it was found that a trial group who had their physiotherapy augmented with 15 minute sessions playing games using the Wii Fit Balance Board, had no significant improvement in objective (such as knee flexion) or subjective (patient perceived level of pain or balancing ability) measures than the control group, who underwent only traditional balance and strengthening exercises. A post trial questionnaire revealed that patients enjoyed taking part in the study. The positive engagement with the video game trial is evidence that even those who may not traditionally play video games can engage with and enjoy a video game based intervention. The lack of improvement over the control group, while better than resulting in a deterioration of condition, points towards the need for video game based rehabilitation to be customised towards a patient's exact needs in order to facilitate a positive outcome.

Similarly, Clark et al. [CBP+10] evaluated the Wii Balance Board as a replacement for more costly balance determination mechanisms, such as force platforms. The work noted that while the Wii Balance Board suffered from a higher *minimum detectable change* (that is, the smallest change in imparted force that will result in a new value being reported) than the clinical standard force platform, the readings were still seen as valuable. The readings were collated from a trial of 30 participants, from two tests at least 24 hours apart. The similarity in result quality raises the notion that the Wii Balance Board could be used to replace basic testing appointments, when used as part of a rehabilitation routine with standardised routines.

In a later trial, Gil-Gomez et al. [GGLAC11] accessed the ability for a Wii Balance Board based system to be used in the rehabilitation of balance problems stemming from acquired brain injury (resulting from trauma or stroke). The reported on *easy Balance Virtual Rehabilitation* (eBaViR) system utilised the Wii Balance Board in conjunction with PC-based software rather than being connected to a Wii; this affords the use of bespoke games specifically designed for medical rehabilitation in collaboration with relevant medical specialists, with associated metrics recorded. As described in the work, the eBaViR system comes with 3 games, allowing for some variety in a play session; each game is prefixed with a calibration stage in which the maximum amount of movement the user is capable of achieving on each axis is determined, so that gameplay can be adjusted to match. Gameplay progress is displayed on screen, and graphical and aural feedback is included as a motivational measure. The trial intervention itself was made up of 17 adults with acute brain injury taking part in 20 sessions of rehabilitation using the eBaViR system, each lasting one hour. Results of the trial indicated improvements in measures such as the Berg Balance Scale and Timed up and Go. Additionally, all of the trial participants reported a positive experience using the system; with a wide age range (mean 47.3 $\pm$ 17.8) is further evidence that age is not a determining factor in the engagement with a video game based rehabilitative routine.

Agmon et al. [APP$^+$11] utilised the Wii Fit and Wii Balance Board in a trial of 7 elderly participants, in a longer term test that required 90 minutes of play time per week for 3 months. Results were determined via the Berg Balance Scale described earlier. Of the seven participants, four had a significant improvement in their BBS post-trial, with participants commenting on increased confidence in stepping and balance, and of their enjoyment in playing the Wii Fit game. Of note is that the trial contained a period wherein patients were supervised by physical therapists until such time that they felt confident in playing the game independently; for a game designed specifically for rehabilitation of patients without supervision, there may be need to adjust the presentation of gameplay, with a long 'lead in' time of more simple gameplay mechanics in conjunction with positive reinforcement before more intensive tasks can be undertaken.

The confidence boost and enjoyment noted in Agmon et al. was also shown in the work of Young et al. [YFBC11], in a trial to determine the efficacy of the Wii Balance Board in improving the standing balance of older adults. The trial consisted of 6 older adults (mean 84 years) playing games using the Wii Balance board over 10 sessions in a month, with all participants reporting enjoyment of the games, and a desire to continue playing. Rather that use the Wii Fit game, two small bespoke

games were developed using the *Virtools* software package (http://www.3dvia.com/products/3dv virtools; *Dassault Systemes*), a 3D environment package with plugin support. In one, the medio-lateral (to the side) control of the centre of pressure recorded by the Wii Balance Board is used to control an on-screen basket, to collect falling apples, while in the second game medio-lateral (sideways) and anterior-posterior (forward / backward) control is required, to move an on-screen character in order to pop bubbles. The bespoke nature of the games afforded the authors the ability to record additional metrics such as item appearance time to file, in order to further analyse patient response to the game. The main result of the trial was a slight, but significant reduction in anterior-posterior sway in the participants post-trial, suggesting that games designed to meet specific needs can have a real impact on the well-being of patients.

The Wii Balance Board and the Wii Fit game have also been utilised outside of elderly balance rehabilitation. Jelsma et al. [JPFJS13] used the device and game in a trial of 14 children with hemiplegic cerebral palsy. The asymmetrical nature of a hemiplegic stroke tends to result in reduced balance, and a reliance on the unaffected side of the body, making medical intervention desirable. Children between 7 and 14 played various minigames from the Wii Fit software in 25 minute sessions 4 times a week for 9 weeks, in conjunction with any standard physiotherapy they were already undertaking. Post-trial results indicated that the gaming based intervention resulted in a statistically significant increase in balance scores taken using the *Bruininks-Oseretsky* test, however several participants had a *degradation* in Timed Up and Down Stairs metric result. As with other trials involving video games, the majority of participants reported that they enjoyed the video game based trial, and the majority preferred it over traditional rehabilitation routines.

2.6 Rehabilitation utilising the Microsoft Kinect

The Kinect motion controlling device released in 2010 by Microsoft for their *Xbox 360* console has also been used for research into rehabilitative gaming. The device itself is a combination of colour and depth cameras, augmented with software designed to detect player movement and represent it as an interconnected 'skeleton' of hierarchical joints. The device and its working will be discussed in more detail in Chapter 3.

Lange et al. [LCS+11] used the Kinect device in conjunction with a bespoke game to asset its usefulness as a rehabilitative tool for those with balance issues stemming from the side effects of stroke and brain injury. The game requires the player to stretch out their hands, controlling an avatar on screen that copies the user's

movement, with the aim of collecting items on screen. Qualitative results from the patient trial were positive, with patients commenting on the physical difficulty of the game, and its addictiveness - a beneficial feature of any rehabilitative game. No quantitative results are presented, however.

CY Chang et al. [CLZ12] compared the Kinect system against an *OptiTrack* motion capture system (http://optitrack.com; *NaturalPoint, Inc.*). The two systems were set up at the same time, with the trial participants wearing reflective markers across their body, for the OptiTrack system to detect them. Participants played a bespoke video game which required them to perform a number of upper-limb motor tasks including shoulder rotation and diagonal body stretches. The results indicated that the Kinect could track linear movements to a similar degree to the commercial motion capture system, but that rotations of the shoulder were poorly tracked - it was posited that this could be due to the Kinect being a single point of view system, and therefore lacked the 3D information required to determine the amount of rotation in an outstretched arm. The OptiTrack system requires additional setup to be performed however, as it tracks reflective markers placed on the body at specific points, such as the joints. The Kinect device is also significantly cheaper, making it more suitable for end users.

YJ Chang et al.[CCH11] utilised the Kinect in a similar way in a trial targeting rehabilitation for children with insufficient muscle endurance. In a trial in which the participants played a Kinect-based rehabilitation game, designed to test whether specific limb movements had improved to therapist determined levels, there were significant changes in performance compared to their baseline performance at the start of the trial. As well as resulting in improved limb movement post-trial, participants found the system interesting enough to wish to keep playing it, and the therapists handling the trial subject's rehabilitation found the system reduced their workload compared to an all-manual rehabilitation, neatly encapsulating the potential benefits of rehabilitative games.

Work undertaken by Dukes et al. [DHHW13] utilised a Kinect in conjunction with a bespoke video game for the rehabilitation of upper limb following hemiplegic stroke. Players of their *Duck Duck Punch* rehabilitation game are required to hit on-screen targets in a carnival-like environment (see Figure 2.11; left). The system as described is not designed to be deployed to a patients home for further intervention without supervision, as the target mechanism is controlled at runtime by a physical therapist via a second screen.
The system utilises a manual calibration step to adjust the properties of the game to suit the unique movement limitations of the player, with the assisting therapist

Figure 2.11: Left: The carnival game environment employed in the work of Dukes et al. Right: Real-time target area adjustment allows a monitoring therapist to target specific areas for the patient to reach for in game, with their motions tracked using the Kinect device

requesting the player to outstretch their paretic arm, and adjusting game properties once they have visually verified user compliance. By determining the user's maximum reach, the equivalent movement of an on-screen virtual arm can be scaled as necessary to match extents on-screen, widening the system's usefulness to a greater range of impairment. Targets are distributed on screen such that some of them are at the edges of the player's reach, encouraging attempts at reach improvement. Areas of the screen can be removed at runtime from the target distribution algorithm by the assisting stroke therapist, utilising the secondary screen (see Figure 2.11; right). No quantitative results are provided from the small trial (6 participants with hemiplegic stroke), but verbal feedback indicated that participants enjoyed playing the game, with higher motivation to play than in normal physical therapy.

The work notes that users had trouble understanding that their arm move the arm on screen, requiring further refinement of the gameplay experience, with the addition of a practice game. While no information as to the mean age of trial participants is presented to gauge likelihood of familiarity of gaming concepts, it is evidence that even what at first thought would be an intuitive control mechanism - a real arm controlling a virtual arm - required adjustment and feedback in order to be effective, demonstrating the need to carefully consider every aspect of the presentation of gameplay.

The full skeletal tracking system employed by the Kinect makes it suitable for a wide range of video game rehabilitation therapies. Sato et al. [SKSN15] investigated the Kinect for its suitability for video game rehabilitation of walking and balance in the elderly. The work describes a pilot study utilising the Kinect's skeletal tracking within a bespoke gaming application with a number of sub games. The work is notable for its full use of the skeletal tracking data provided by the Kinect - The balloon popping minigame (Figure 2.12; Lower Left) focuses on utilises the Kinect's ability to determine 3D joint positions to calculate the angle of the knees in order to detect whether the player is correctly performing a squatting manoeuvre, while the one-legged-standing game (Figure 2.12; Lower Right) uses knee and hip

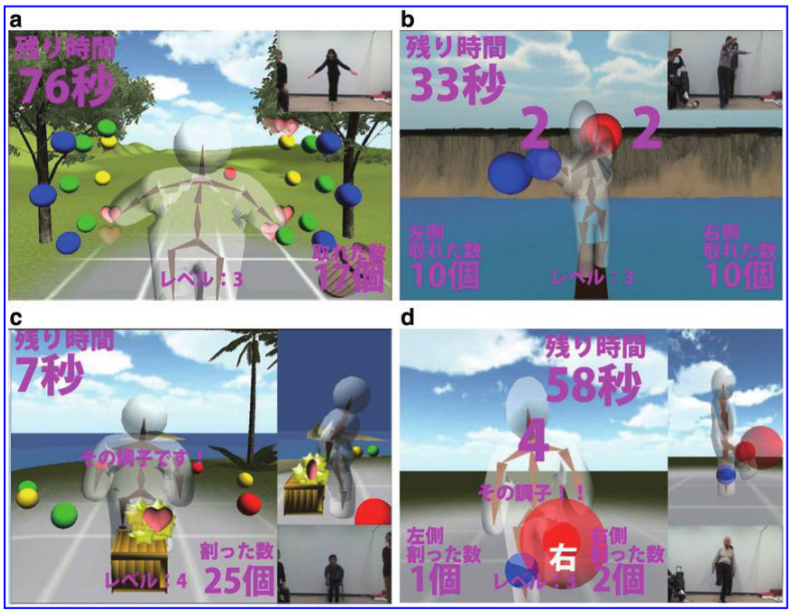


Figure 2.12: The bespoke minigames outlined in Sato et al. in which the Kinect's skeletal tracking was used to detect the correctness of a variety of poses, including knee bending and standing on one leg

placements to determine that the user is posed correctly. Visual feedback is provided as appropriate to ensure player compliance with the rehabilitative goals of the game.

Of the tracking devices commonly used with home video game hardware, only the Kinect would be capable of providing a reliable calculation of knee usage; force sensing devices such as the Wii Balance Board cannot accurately determine whether a shift in balance is from leg or arm movement, demonstrating the unique capability of the Kinect for certain rehabilitative moves. In a trial of 28 participants over 24 game sessions, a significant slight improvement in mean Berg Balance Scale was reported (55.31 pre, 55.97 post), and significant increase in Functional Reach; neither increase was noted in the control group. The positive results suggest that utilising all of the data available from an input device, in this case full skeletal tracking and joint angle determination, encoding a range of motions in a gameplay environment, and providing meaningful feedback as to how to succeed at the gameplay, can lead to meaningful improvements to standard measurements of ability.

The depth sensing capability of the Kinect system has been put to use in cognitive therapy, as well as physical therapy. The work of Gonzlez-Ortega et al. [GODPMZAR14] utilises the Kinect to augment the INTRAS software platform designed for the rehabilitation of body scheme dysfunctions (inability to intuit how to order one's limbs into a pose) and left-right confusion. The software presents a number of exercises to the user, based around asking the user to touch part of their body, and then detecting success via a camera. Utilising basic 2D cameras presents a detection problem - although it can be determined that something has covered the

body part to be touched (such as the left ear, or right eyebrow), the lack of depth information makes it harder to reliably detect that the occluding object (assumed to be the user's hand) is directly *touching* the desired body part, or simply occluding it from the camera's viewpoint. The addition of a depth-image and skeletal tracking to the Kinect affords greater reasoning of exact body positioning; it can therefore be determined that it is the desired hand covering the body part, and that the hand is within a depth bound. While no comparison results with the system prior to Kinect integration are presented, results from a trial 15 participants performing 14 exercises across 5 sessions revealed a successful gesture detection rate of 96.28% accuracy.

2.6.1 Rehabilitation Using Playstation Move, PSEye

The rising popularity of inexpensive gaming devices that can detect the motion of the user has also introduced many new avenues for research into rehabilitative gaming. The work of Flynn et al.[FPB07] focuses on the usage of the *Eyetoy* webcam peripheral for the Sony PlayStation 2. Over the course of 20 unsupervised hour long sessions of playing a commercial video game title designed for the Eyetoy, a ~10% increase in Fugl-Meyer Assessment score was attained by the trial subject, who was motivated to continue gaming based rehabilitation, demonstrating that even 'off the shelf' gaming hardware and software otherwise not designed for rehabilitative purposes can result in meaningful improvement. Similar work by Jannink et al. [JvdWN$^+$08] in which a trial was performed with children with cerebral palsy resulted in improvements in medical assessment scores after playing the EyeToy: Play game for a 6 week period. No negative side effects were reported, but it was noted that the off-the-shelf nature of the hardware and software being utilised made it difficult to create the right conditions for maximum therapeutic benefit, and that it would therefore be beneficial to be able to adjust difficulty, and the types of feedback presented by such software.

2.6.2 Rehabilitation with Other Devices

The work of Bardorfer et al. [BM01] explores the usage of a novel handheld haptic interface as a means to both input movement into a simple maze game, and as a means of providing feedback to the user by exerting force to indicate to the player that they are veering off course. The PHANTOM (www.dentsable.com/haptic-phantom-omni.htm; *Geomagic Inc.*) haptic device takes the form of a pen like device attached to an armature, that allows the user's manipulation of the pen to be recorded to a stated accuracy of 0.03mm in 6 degrees of freedom, while additionally being able to provide 'force feedback' to oppose the user's movement up to 8.5 newtons. The work describes a test where data recordings are made of a player using the PHANTOM device to move a ball through a 2D maze, with

enough fidelity to detect details such as an increase in hand tremor over time. The simple task and focussed, high quality of data output made available by the described system affords repeatable, quantifiable measures of change in performance over time, a keystone in the concept of a self-contained rehabilitative video game. The later work described in [LFR09], where a similar haptic device, the *Novint Falcon* (www.novint.com/index.php/novintfalcon; *Novint Technologies Inc.*), was used for rehabilitative properties showed that even compared to the freedom of advanced motion controllers such as the Wii Remote, haptic interfaces can still provide beneficial results, with the force feedback capability of the described device making several trial participants feel like the games were as physically exhausting as traditional rehabilitation routines.

There has been interest in using haptic 'gloves' for hand rehabilitation; these are devices capable of determining the amount of flexion in each digit of the hand. Morrow et al. [MDBM06] exams the usage of the Essential Reality P5 glove for the purposes of stroke rehabilitation, in conjunction with a modified XBox console. Several games developed in Java are described, designed to test speed and range of finger motion - the user may be tasked with catching butterflies, or sweeping away dirt from an image. While no trials using the system were described, the work does reiterate the concerns noted throughout the literature that devices designed for fully able-bodied individuals may not be suitable for use by people with physical and neurological disabilities that affect the limb.

The CyberGlove (www.cyberglovesystems.com/cyberglove-ii; *CyberGlove Systems Inc.*) range of commercial finger tracking devices has been the focus of a number of works related to rehabilitation. Jack et al. [JBM$^+$01] and Merians et al. [MJB$^+$02] utilised the CyberGlove in conjunction with an additional force feedback glove to conduct a clinical trial to establish the usefulness of the combination for rehabilitation of stroke. A number of simple software exercise are presented, designed to improve the user's finger speed and strength, along with their range of motion. The trial software was designed such that failure to complete a task made further attempts easier to complete. The work notes that the number of game 'levels' undertaken per play session had to be manually reduced to prevent fatigue, something that rehabilitation software must take into consideration to prevent injury. That this reduction in performance could be noticed within the data output of the software is an encouraging sign as to the potential sensitivity of a software solution.

The work of Kuttuva et al. [KBM$^+$06] details work undertaken at Rutgers University, resulting in the 'Rutgers Arm', a device designed for upper limb rehabilitation. The device utilises a tracking device attached to the user's wrist to detect movement

on a specially-designed table, that the player must rest their arm upon. As with other rehabilitative works, the device must first undergo a calibration procedure, to determine the amount of movement the user has. The work describes the use of the Rutgers arm to play a 'Breakout' clone, with the player moving their arm left and right to move the game's 'paddle' in order to bounce balls into bricks. To determine the usability of the system, a single patient trial was undertaken, taking the form of 16 sessions. Data recorded from the trial indicated a significant increase in wrist motion and peak arm movement velocity, and Fugl-Meyer tests performed before and after the trial indicated a 7 point increase.

The Rutgers Arm was further developed, resulting in the work described by Burdea et al.[BCM+08], in which a Rutgers Arm setup was modified to allow the table to tilt, affording the ability to apply some amount of gravity loading to the arm movements. The improved system was used in a trial of 3 post-stroke patients, who played a number of Java-based games using the arm over 12 sessions of increasing duration and intensity. Results indicated an improvement in measurements testing fine motor control post-trial, and an overall engagement with the 'virtual' therapy. One participant found the 'Breakout' game to be 'too fast' at the default speeds, requiring bespoke adjustment, an indicator as to the need for allowing a degree of customisability in a video game delivered therapy.

Further research into related technology at Rutgers led to the development of the Master II force feedback glove, detailed by Bouzit et al[BBPB02]. The glove takes the form of a number of pistons attached to the fingers via ring-like clamps, that can both impart force, or record the force acted upon them, to a level of up to 50 Newtons. Flexion of the fingers is determined via a magnet and Hall-effect sensor placed within each piston assembly, allowing a model of the finger's movement to be determined via inverse kinematics. [AMB+04] describes results utilising the Rutgers Master II glove, as part of a post-stroke rehabilitation routine. A number of game-based exercises was developed to use with the glove, whereby the user was required to reach and grasp items using a virtual representation of their hand on-screen, or 'play' an on-screen piano. Results after the 13 week trial indicated a significant improvement in range of motion and individual finger motion in the majority of cases, and an improvement in task completion time. This demonstrates that there is a potential benefit to utilising haptic and force-sensing devices in stroke therapy, as improvements in finger strength and dexterity may allow for increased independence and quality of life.

The work of [GMW+10] uses a customised glove with position sensing hardware installed within its fingertips, in order to detect changes in finger position. When

Figure 2.13: The *Circus Challenge* Stroke Rehabilitation Game: On screen character performs the limb movements you must replicate to succeed

combined with a number of simple Java games designed to be controlled via simple movements, it was shown that, after a trial periods of playing games for 20-30 minutes daily for an average 20 hours of gametime, the trial participants use of their plegic limb was improved.

2.6.3 Modern Stroke Rehabilitation: Circus Challenge

Comparisons have been made between the CAHAI assessment and a motion controlled video game designed for hemiplegic stroke therapy [SME] [SME14]. The *Circus Challenge* game utilises the *Sixense TrueMotion* (www.sixense.com; *Sixense Entertainment, Inc.*) to determine the 3D position and orientation of the user's hands - a full description of the device is provided in Chapter 3 of this thesis. The game consists of a number game tasks to complete, split up into three difficulty levels. Each task will be in one of a number of different minigame types, with 10 levels per gametype per difficulty. In some minigames, the player is required to replicate motions pertaining directly to the theme of the game; for example in 'Knife Throwing', the user must hold their arm up, and bring it forward, to produce an approximation of the on screen character throwing knives at a board. In others, the user must successfully replicate moves of increasing difficulty, causing the circus character on screen to successfully perform their act- Figure 2.13 shows the on-screen 'juggler' character, who will either successfully perform a juggling trick, or drop her juggling balls, depending on whether the player can replicate the move. During gameplay, the player is shown which move to perform by an on-screen character performing the move to be completed, with colour coded hands to aid understanding (as shown in the bottom right of Figure 2.13). Moves begin with simple movements, such as raising arms straight up in the air, or outwards in an inverted 'V' shape. More

complex moves are introduced throughout gameplay, requiring more complex limb coordination and joint rotation - the player may be required to complete a facsimile of performing a chopping action as if holding a knife, for example.

In addition to the main game, Circus Challenge contains an 'assessment' mode, designed to be played periodically throughout a prescribed therapy. In this mode, the player is guided through a series of 40 preset movements, selected based on their ability to provide a detailed view as to the overall ability of the player at that point in time. The assessment takes approximately 20 minutes to complete, with no retries of failed moves allowed. Internal state metrics and the position and orientation of the controllers throughout the assessment are saved to file, and it is this data that has been analysed for its validity as a replacement for the CAHAI in video games for stroke rehabilitation.

In Shi et al. [SME] this assessment mode was studied using a trial of 26 stroke survivors split between chronic and acute, and comparing it against a reduced form of the CAHAI, consisting of the first 10 tasks. Across 8 weekly assessments of the game and CAHAI, the work concludes that it is possible to derive a statistical model from the assessment output that shows a high correlation with CAHAI scores. This provides evidence that it is possible to produce a medically valid indicator of therapy progress using commercially available hardware in the home. The tight correlation between CAHAI scores and fitted results from Circus Challenge suggest that it is possible to gauge upper limb motor function without the use of props, reinforcing the idea that a piece of therapeutic software can both engage a player, and assess their improvement in a natural manner.

2.7 Detection of Player Movements

Much of a patient's physical rehabilitation will take the form of repetitive simple movements to restore and build up limb strength. However, many of the actions used in assessment of someone's ability are facsimiles of a functional movement - turning a key as in the Wolf Motor Function Assessment, or the ability to pour a glass of water as used in the CAHAI assessment. Such movements require complex limb coordination and dexterity, with possibly quite small movements. This poses a challenge for the creation of integrated medical-quality feedback within a video game rehabilitation system, as while simple poses can be detected from a motion sensing device with relative ease using the reported orientation and position, complex movements require tracking the change of position and orientation over time, possibly of both limbs simultaneously.

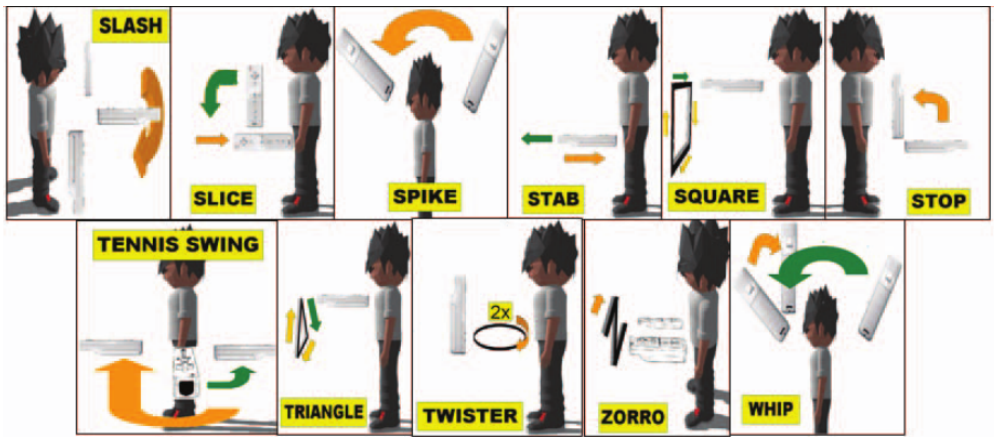


Figure 2.14: An example of the gestures used in Hoffman et al. which are detected via accelerometer data from a Wiimote device, and a machine learning algorithm to select the most likely move from a training set

Hoffman et al. [HVL10] examines two common machine learning methods for their ability to successfully detect a set amount of 3D gestures, a selection of which are shown in Figure 2.14. These gestures were trained into the machine learning tools by collecting gesture attempts from 17 participants. The Nintendo Wii was used, with the acceleration and orientation data being recorded. Using the result of the machine learning training on the original dataset, both of the machine learning methods were able to correctly identify the gestures to an accuracy greater than 95%. The participants in the experiment were all able-bodied, however, making it hard to gauge how such a system would cope with someone who is perhaps incapable of performing the gesture in the ideal way. The work also notes that several of the gestures could not be accurately differentiated by either method, requiring their removal from the system, an indicator as to the difficulties even trained machine learning can have differentiating similar, but distinct movements.

A similar machine learning mechanism is described by Liu et al. [LZWV09], in which the *uWave* algorithm, designed to require only accelerometer data, is introduced. The algorithm works by quantising acceleration data to a range of 33 unique values per axis, and summing the difference in quantised accelerometer and trained data sets across the whole time series, with a gesture passing or failing based on a threshold. The accuracy of the system was high (approximately 97%) when comparing against training data recorded the same day as the test, but dropped to 88% when taking into consideration training data from 6 days earlier. This highlights the major problem facing gesture detection for rehabilitation: minor changes in how the user inputs the gesture may lead to an inability to determine the movement being attempted.

A popular method by which gestures are detected is the *Hidden Markov* model, a method by which a system progresses through a number of possible states based

on a probability distribution. Repeated observations of some external state (such as the state of an accelerometer) are made, and probabilities of one such state leading to another are trained via repeated measurements of what is being observed - in this context, by repeatedly performing a gesture. By increasing the number of unique states within the system, the ability to correctly determine the gesture being performed increases.

Kratz et al. [KSL07] utilises a Hidden Markov Model in conjunction with Wii Remote accelerometer data to determine the gestures being performed by the user in a spell casting game. The HMM for each spell the player can cast was trained by having 7 different users perform each gesture over 40 times. Gestures generally took the form of gross movements in 2D. The system was capable of recognising the gesture being performed over 90% of the time when split into 10 states. The work notes that in cases where the model was not trained using a particular player's data, that the system was only capable of correctly identifying that player's gestures approximately 50% of the time, exemplifying the need for a system to be capable of adapting to the person using it.

2.8 Discussion

Video game therapies have the chance at reducing the social stigma attached to traditional rehabilitation therapies: engagement with the game can be seen as trying to beat high scores or earn achievements, not get better because the player is 'unwell'. Video games are an inclusive entertainment, and a therapeutic game should embrace this; entertainment (that is, non medical) games often support competitive or cooperative multiplayer, or multiple accounts allowing multiple users to play, and there is no reason why this should not be the case even in a more medically oriented game. Family interaction should be encouraged, as becoming part of an enjoyable daily routine will increase compliance with the underlying medical applications of the product.

Care must be taken when prescribing video game rehabilitations, as games have been linked to an increase in seizures [FHE$^+$05] in people susceptible to them. Overindulgence in motion controlled video games such as Wii Sports has been reported to lead to strain injuries [Bon07]. It may be that the rehabilitative game itself should be aware of such concerns, and balance in-game exercises across limbs and muscle groups as appropriate, or even refuse to continue play if used too much in a single session.

The literature presents an insight into the wide range of ways in which physical

and neurological disabilities can manifest themselves. For this reason, it seems intuitive to reason that no single game design will map exactly to the needs of any one patient; therefore, some amount of customisation is desirable [ALMK10a].

The small sizes of early trials investigating the efficacy of video game therapies for ailments such as stroke has made it difficult to gauge how useful 'virtual' rehabilitations would be when applied to the wider population of those requiring them [CLBM07]. Since then, the introduction of peripherals that can accurately detect position and movement has vastly increased the amount of research literature pertaining to video game rehabilitation. The increasing popularity of games and gaming peripherals designed around promoting health benefits has led to increased regulation and official guidance, to ensure that they are held to a medical standard where necessary[Foo13]. This reinforces the need for rehabilitative software to be able to provide accurate, and medically verified information so as to best inform both patients and therapists as to how best to proceed with meeting therapeutic goals.

The Nintendo Wii has been a phenomenally successful console, and with its motion sensing controllers and popular games utilising them, was a natural fit for health rehabilitation research. The literature shows that Nintendo's own Wii Sports and Wii Fit software can be an effective adjunct to ongoing therapy in a number of different areas. The work of Mouawad et al. [MDMM11] indicates that the added physical activity due to playing Wii Sports regularly resulted in improvements to traditional measurements of post-stroke ability. The results of Saposnik et al. [STM$^+$10], indicate that it is the physical activity, rather than simply the engagement in the task that results in the improvements, as primarily mental activities such as board games resulted in no such improvement, even if described as enjoyable.

While the Wii Fit game was designed and marketed towards those looking to maintain a healthy lifestyle, its focus on accurate balancing and shifting of the user's centre of gravity has led to its use in research in the rehabilitation of a range of ailments, to varying degrees of success. The Balance Board controlled game is enjoyed by its users, young and old [JPFJS13] [YFBC11]. Medical benefits of the device have been slight, but significant - The Berg Balance Scale and Timed Up and Go improvements of Gil-Gomez et al. [GGLAC11] Sugmarman et al [SWEBB09] and Young et al. [YFBC11] point to the ability of rehabilitation games to show real-life improvements. However, the results discussed in Fung et al. [FHS$^+$12], where post operative knee-replacement patients displayed no improvement when using the game regularly suggests that the types of exercise within the game may not be beneficial to all, requiring research into what types of therapy can be encoded within

a game. It is important to note that even though there was no measurable benefit, all patients enjoyed the game, suggesting that games are still a useful inroad for exposing people to rehabilitative therapy.

The research conducted into the efficacy of Dance Dance Revolution for rehabilitative gaming has shown that even a game designed without direct medical measurement or monitoring of the user can have a positive impact on those playing them. Works such as [MBK+08] and [MCN+09] have shown that an appropriately designed game can engage a player over multiple gaming sessions over the course of many weeks, with results that indicate a trend towards lifestyle improvements (reduced sedentary time, reduced weight gain compared to peers). While research has primarily targeted the problem of excessive weight in children, the game has shown itself capable of integration into therapy into neurological disorders, as in Kloos et al. [KFK+13], whereby post-trial analysis revealed significant improvements in ability to support oneself and walk forward in those afflicted by Huntington's disease. Works such as [SSS+09] and [SPH+10] show that even the elderly can enjoy a DDR style game, if sufficient considerations are made towards their reduced abilities, and that successful engagement with rehabilitative software can improve self-reported balance confidence and mental health. If games can improve people's perception in their own abilities, then this must be seen as a bonus even if no statistically relevant changes to existing metrics are made as a result of playing the games.

Of the works discussed in this chapter, only the Circus Challenge game investigated in Shi et al. [SME] [SME14] has been designed with the aim of mapping to existing therapeutic success measures. The concept of mapping in-game performance to existing success metrics should be focussed on in future health gaming research, as such a feature would allow such games to be used as a more engaging replacement for such measures, while providing well understood metrics familiar to therapists. Ease of use by therapists is important for health games to be a viable alternative to traditional therapies, as they will likely be the people instructing patients as to how to use them, and investigating their progress; this ease of use must include both the analysis of progress, but also in the ease in which the software can be deployed. Bespoke solutions with custom modified hardware such as [MDBM06] may prove to be difficult to work with due to the unique nature of the devices.

The nature of many of the rehabilitation games discussed in this chapter, that is, custom-designed software built in an academic institution, may limit the uptake rate of the software; while perfunctory graphics and sound are acceptable in software designed for research purposes, games developers will spend a great deal of

time and money into assessing the usability of their games, with focus on the user experience 'feel' of the software, such as how easy menus are to navigate, how intuitive the controls are, and ensuring the game remains a rewarding experience after many plays. Rather than developing small 'rehab research' games in-house, it may be better to instead investigate the means by which rehabilitative exercises can be inserted into the games being produced by the games industry, allowing the years of expertise at creating successful products be harnessed, while also allowing therapist input into the design of the therapeutic aspect of the software.

It is clear from the literature discussed here that video games can be of benefit to those requiring some sort of physical therapy. At the most basic level, a well designed game provides enjoyment, which several of the works note as an important factor in maintaining quality of life. The physical nature of the games, requiring the use of motion or balance sensing devices, combined with the enjoyment factor, allows for those requiring physical therapy the means by which to remain active, and more actively engage with their therapy. In terms of matching the desired outcomes of traditional therapy, more research will be needed to determine how best to combine therapy and gameplay. The presented works do note statistically significant improvements to several common metrics, suggesting they are an effective means by which to provide therapy, but as the trials generally work as an adjunct to existing therapy, it remains to be seen whether a well designed video game can effectively replace traditional physiotherapy in the home.

2.9 Conclusions

The literature presents a clear indication as to the potential benefits that video game technology can bring to the rehabilitation and alleviation of the symptoms of physical and neurological injury. The introduction of motion and force detecting devices has led to a body of research that uses these cost effective devices to aid patients with statistically significant results. Many of the sources make a point of showing how patients engaged with the game software to a level beyond that of traditional exercise and rehabilitation routines; it is common for such interventions to suffer from high failure rate due to lack of motivation in completing assigned tasks, making gamification of such tasks beneficial.

A common theme within the literature is the requirement for a game session to be supervised to ensure that the player is performing tasks correctly, and to instruct the player to do certain things in accordance with the research goals of the project, which may not match up to the on-screen prompts and natural gameplay 'flow' of the software. This opens questions as to how best to produce a piece of health soft-

ware that operates to the maximal benefit of the user without necessarily requiring external assistance.

Of the rehabilitative gaming trials discussed, relatively few of them have used bespoke software, mainly relying on off-the-shelf games such as Wii Sports. This limits the quantitative analysis that can be done, as any in game-results and scoring are unlikely to be based on medically approved criteria such as the CAHAI scale, and therapist judgement on game performance is subjective. Despite this, evidence does point to the potential for video games to make a meaningful improvement to people taking part in rehabilitation for physical impairments.

In order to support the physical rehabilitation of patients, it will be necessary for software designed around this concept to be able to determine and measure the player's ability to the complete the rehabilitative motions required of them, both as a means to guide the player and determine their progress through the game, but also replicate the standardised tests traditionally used by therapists, and thus provide a common means by which real-world changes in ability can be measured. While there is literature pertaining to the detection of gestures in 3D space, they tend to be used in a different manner - rather than detecting the quality of a movement being performed, they instead determine which gesture out of a pool of possibilities is being performed. This difference leads to drawbacks when trying to apply existing techniques to physical rehabilitation, as the differing mechanisms that determine which move is being performed don't necessarily lend themselves to providing meaningful feedback as to how well a move was performed, or where the move attempt failed in cases where the user incorrectly attempts a movement.

As noted in works such as Hoffman et al. [HVL10], machine learning algorithms require training in order to be capable of detecting a particular gesture. Such an approach may prove problematic for rehabilitative purposes as there is the possibility that the user cannot successfully complete all of a move, resulting in an incomplete dataset from which to attempt to determine the gesture being performed. A patient's ability may differ substantially from that of the person performing the initial training, causing further issues.

Ideally, a method of determining 3D movements would not return a binary pass or fail result, but be able of determining where an unsuccessful move attempt deviated, so as to provide relevant feedback to the player and therapist as to how best improve. Such a system must too be able of adapting both to the unique geometry of the user's limbs, and to the changes in their ability over time, as the overall area that a user may reliably move their limb in 3D space during a gesture attempt at the

start of therapy may differ a great deal to that of an attempt when a limb's dexterity and range of motion has improved. Any change in the user's ability to replicate a movement, whether positive or negative, is of importance to therapists, so accurate detection of how a gesture is performed is necessary - this may involve splitting gestures into discrete sections that can be tested and reported on separately, allowing therapists to infer which types of motion are resulting in noticeable changes to dexterity. A patient with severe motion impairment being able to lift their impaired limb 20cm can be seen as a 'success', so it may be necessary for gesture attempts to take into consideration the overall reachability of the user, with this too being a metric that may indicate a positive change in condition over time.

While many of the devices designed specifically for medical use, such as the Rutgers Arm, are specifically tested for the accuracy of their input, there is relatively little research into the ability for commercial devices to accurately gauge the user's movements; instead, work involving these devices generally focuses on simply getting the player involved and engaged in their physical therapy. For a rehabilitative software package to play a useful role in a person's physical therapy, it must be capable of providing meaningful feedback as to their physical ability to both the player and their therapists. It must therefore be designed such that the types of movements being targeted be tested to a high degree of accuracy over time, so as to allow for accurate conclusions as to the player's progress be made. Upper limb therapy may involve both gross muscle movements and more finer grade motions requiring dexterity and fineness of touch; therefore an ideal physical rehabilitation package should engage with and measure both the user's ability to move their affected limb, and their ability to impart a varying amount of force, to a high level of accuracy.

The next chapters of this thesis aim to explore possible solutions to these issues. Chapter 3 seeks to measure the accuracy of a number of commercial motion and force sensing devices, in order to determine their applicability in recording player motions to the fidelity necessary to reason about their performance in rehabilitative games to a level similar to that of traditional performance metrics. Chapter 4 continues by investigating the use of force sensing devices to determine the best methods by which to promote positive outcomes from physical rehabilitation, while Chapter 5 will investigate the use of motion sensing devices to detect the player's ability to replicate the gestures set out for them by a therapist.

Chapter 3

Accuracy Metrics Of Input Devices For Stroke Rehabilitation

3.1 Introduction

The literature described in Chapter 2 outlines work utilising a wide array of input devices to track a patient's movement, to aid in the rehabilitation of a wide range of ailments. While some, such as the Rutgers Arm [BBPB02] are bespoke devices, or commercial devices that have since ceased production [MDBM06], there is an increasing body of work focussing on currently available input devices tied to popular gaming consoles: the Nintendo Wii, the Playstation Move, and the Microsoft Kinect. Each of these devices has proven to be incredibly popular, and are widely available at affordable prices, making them ideal targets for video game rehabilitation in the home. All of these devices can be used in some way to track the user's hand position and orientation, with the Kinect having the added ability to determine the joint orientation and positions of a virtual 'skeleton' matching the user. The resurgence in popularity of Virtual Reality using headsets such as the *Oculus Rift* (www.oculus.com/en-us/rift; *Oculus VR, LLC*), has led to commercial products designed to facilitate accurate tracking of the hands, such as the Sixense TrueMotion, and the Leap Motion devices (www.leapmotion.com; *Leap Motion, Inc*).

In order for their use in rehabilitation software to be viable, such devices must provide accurate, low latency, and low error results. While there has been some work on determining the accuracy of the Kinect's skeletal tracking ability [FBSL12], there has been little in the way of determining the accuracy of the Wii Remote or Playstation Move, or any comparison of relative merit between the devices, or between more recent hardware developments such as the Sixense device.

As part of a home based therapy, it may be necessary to measure the amount of strength in a user's hand, for example to determine whether it is safe to introduce

props into the therapy, in a manner similar to that as the Wolf Motor Function scale, or the Fugl-Meyer Assessment. It is also the case that for those suffering from acute stroke, that the muscle weakening side effects of hemiparesis mean that holding and moving motion devices such as the Wii Remote and PS Move is difficult or impossible, and therefore alternative methods of enabling interactive therapy should be investigated. For such users, it may be more beneficial to focus their interactive therapy on building strength and coordination in their hand via the application of force, in a manner broadly similar to that of the Wii Balance Board, but focussed on just their affected limbs.

The purpose of this chapter is to investigate the accuracy of a number of motion and force sensing devices, to determine their suitability as input devices for rehabilitative products for stroke. The devices utilise a number of different mechanisms to capture user motion, ranging from RGB and depth cameras, to magnetic field measurement. A common benchmark is proposed and tested using a number of motion detecting devices, with the sensitivity of a number of force sensing devices also tested.

3.1.0.1 Research Goal

This work constitutes an advancement in the understanding of the current state of the art in commercial motion sensing devices, and serves as a starting point for those looking to determine which motion devices to use when creating rehabilitation software for stroke. A measurement of overall motion quality is proposed, and example testing procedures outlined. Within the context of this thesis, the work contained in this chapter informed the work on acute stroke input detection discussed in Chapter 4, and gross movement detection detailed in Chapter 5.

The work described in this chapter pertaining to motion sensing devices was presented under the title *Benchmarking Motion Sensing Devices for Rehabilitative Gaming* at the *30th ACM/SIGAPP Symposium On Applied Computing*.

3.2 Accuracy Metrics Of Motion Sensing Devices

In this section, 5 commercial motion sensing devices are investigated for their capability to provide motion tracking suitable for use in rehabilitative games for sufferers of stroke. Current sensing hardware is not perfect - environmental factors such as temperature and pressure can result in signal deviations from the true value being detected, as can physical limitations of the device; a sensing system based on digital camera technology will be limited by pixel density and colour dynamic range, for instance.

More generally, a sensor can be thought of as returning a signal at a given frequency, and within a given range. Deviations in this signal from the true value affect the overall quality of the signal output, and should be minimised. These deviations can be broadly classified as being either delays in reporting changes to the value being monitored, or in low and high frequency *drifts* from the true value over time [AEL+00], with the amount of drift possibly varying or wavering [CM07], making it difficult to detect and remove.

To be suitable for accurately detecting player movement, a device should minimize both latency and signal error over time. This is both to ensure the accuracy of the data being recorded, and in ensuring that the game reacts in a timely and accurate manner; accurate representation of player movements despite signal latency and bias is a major goal of hardware developers of gaming peripherals [VR16]. To determine the suitability of the chosen motion sensing devices for rehabilitative purposes, they have each been investigated for their susceptibility to low and high frequency noise, in addition to latency.

3.2.1 Latency

In order for rehabilitative software to make reasoned judgements as to a user's movements, a motion tracking device needs to output new information at a high rate. Motion devices connected to a PC using the standard *Universal Serial Bus* are polled by the host machine for information at a rate of 125Hz (*Hertz* being the standard metric for measurement of periodic events, with 1Hz being one event per second), meaning that data returned from polling could have been recorded by the device up to 8 milliseconds previously. This is a best case, however, as most of the devices discussed here update at a far lower rate; for example the Kinect camera runs at a rate of 30Hz, so that no matter if the host machine polls faster, there will only be new picture data provided from which the provided device library can derive position from once every 30th of a second. While such an update rate is more than enough for the perception of fluid motion in visual medium such as film, it may be too low to accurately detect high-frequency oscillations in physical movement. Therefore, update rate is an important factor when investigating motion devices.

3.2.2 Drift

Signal drift from the true value can take place in both low and high frequency domains. High frequency drift is generally known as *noise*. One method of determining the level of noise in a sensor signal is to measure the signal at a base line level - in the case of the sensors being tested, base line noise can be seen as the orientation

and positional changes recorded by the device while still. Such 'background' noise in a signal is almost impossible to avoid, but devices that exhibit large variations in high frequency drift can 'mask' a user's real movement, and therefore a device should be investigated for its noise level.

The tested sensors should also exhibit minimal amounts of low frequency noise. Such deviations in sensor value are often caused by environmental factors such as temperature change or the effects of stress upon a sensor subjected to physical force. These factors result in a gradual change in baseline reading that must be accounted for. Whereas noise describes an oscillation from the 'true' value, this low frequency *bias* describes a constant difference from the true value. For example, some accelerometer units utilise a small metal mass held in place by springs, with a value reported utilising the Hall effect [nih13]. The mechanical nature of the springs can cause small deviations due to temperature, or due to extreme movement; a quick shake may result in the metal mass settling at a slightly position than before, resulting in a small amount of bias. The nature of the signal bias, if any, reported by a motion device should be investigated, as an unpredictable amount of bias will be difficult to correct for, and will result in inaccurate measurements.

Also of concern when investigating the quality of data output in devices is 'hard error', a term used by the author to describe situations where tracking is lost, and how the device deals with this. Camera based devices are particularly susceptible to hard error, as the object the device is visually tracking may become obscured. As this is impossible to entirely avoid, how quickly a device will recover from a hard error situation, and its behaviour during this time, should be assessed.

3.3 Overview of tested devices

3.3.1 Wii MotionPlus

The Wii Remote is a small hand held controller (shown in Figure 3.1, left), consisting of a number of analogue and digital buttons, similar to other modern console controllers. Unlike the controllers bundled with other consoles of the time period, the 'Wiimote' additionally contains an three-axis accelerometer, allowing the device to detect the direction of gross hand movements. Due to Wii games promoting active movement of the device, the Wii Remote also has a lanyard strap, to tie the device to the user's hand to prevent damage caused by losing grip of the Wii remote and accidentally throwing the device.

In addition to the accelerometer, the Wii Remote contains a forward facing infra-red camera, with a resolution of 128x96. This camera is used in conjunction with the

Figure 3.1: Left: The Wii Remote device. Right: The Wii Sensor Bar, which contains infra-red LEDs that are used to aid in orientation and z-axis tracking

Wii 'Sensor Bar' (Figure 3.1, right), an enclosed unit containing infra-red emitting lights placed 20cm apart, designed to be placed above or below the user's television. The Wiimote supports an add-on 'MotionPlus' device that can be connected to the back of the controller, containing gyroscopes, affording rotational information to games supporting it. Later versions of the Wiimote has the 'MotionPlus' unit built directly into the housing.

As there is as yet no official method of communicating with a Wii Remote on a platform other than the Wii itself, there is a reliance on third party drivers. For the purposes of this chapter, the *Wiiuse* library was used (www.github.com/rpavlik/wiiuse; *Ryan Pavlik*) to allow for acceleration, gyroscope, and IR image communication via the wireless *Bluetooth* protocol (www.bluetooth.com; *Bluetooth SIG, Inc.*).

3.3.2 Limitations of Accelerometer

It is intuitive to think that having access to the data from a 3-axis accelerometer is sufficient to derive a position from, using a double integration over time to calculate both velocity and position, as in Equation 3.1:

$$P = \int \hat{v} \cdot dt = \iint \hat{a} \cdot dt^2 \tag{3.1}$$

This method is prone to inaccuracy, however. The accelerometer data from commercial-grade accelerometers such as those in the Wiimote suffer from some degree of both noise and bias [Ana06], which when integrated over time quickly leads to inaccuracy, in the order of many metres per second additional perceived movement. The accelerometer in the Wiimote is only rated to detect up to 3G of movement [Ana06], whereas a person may move their arm with a peak acceleration of beyond 11G [Nag89]; this will have the effect of producing an inaccurate signal at high movement speeds, further distorting the final position calculation.

Figure 3.2: An estimation of orientation can be made using the effects of gravity on the accelerometer (Left) , however orientation cannot be determined around the gravity axis (Right)

While unsuitable for measuring an accurate 3D position, the accelerometer does allow for limited detection of the device's orientation. The accelerometer is affected by the force of gravity, meaning that when held stationary, it can be determined which direction has acceleration still enacted upon it, and can therefore be considered 'down' from the user's perspective. By treating the values retrieved from the 3-axis accelerometer as a vector, a simple normalisation operation can be used to determine the downward direction. Without the gyroscopic data from the MotionPlus peripheral however, it is impossible to determine orientation *around* this downward axis. Figure 3.2 demonstrates this further: on the left, it can be seen how the effect of gravity maps from the y axis to the z axis as the device tilts upward, affording a basis for orientation, while the right image shows how rotation around the gravitational axis cannot be determined, as the mapping does not change.

3.3.3 Camera based position detection

As the accelerometer utilised in the Wii is unsuitable for determining accurate position, alternatives must be investigated. The position of the Wiimote is usually determined using a combination of the infra-red camera at the front of the Wiimote unit, and the light sensor bar. The camera sees two 'blobs' of light emitted from the light sensor bar, allowing for the determination of several metrics. By processing the positions of the blobs in the camera image, it is possible to determine the rotation around the z axis of the device (that is, the amount of *roll*), using the two argument arctangent function:

$$\begin{aligned} v &= (a_x - b_x, a_y - b_y) \\ \hat{v} &= \frac{v}{||v||} \\ roll &= atan2(\hat{v}_y, \hat{v}_x) \end{aligned} \tag{3.2}$$

where x_1 and y_1 are the coordinates of the 'left' blob of IR light, and x_2,y_2 the coordinates of the right IR blob.

The user's distance from the sensor bar can also be determined using the IR camera. As the sensor bar emits two IR sources, at a fixed distance apart of 20cm, the user's relative position can be determined from the distance between the light sources in the camera image, and the field of vision of the camera, using the following equation:

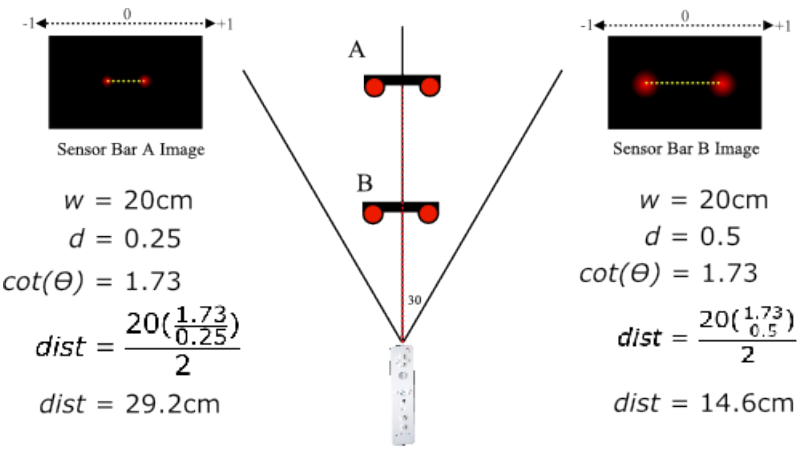


Figure 3.3: Distance determination using the Wii sensor Bar. The distance between the detected light positions is proportional to the distance between the device and the sensor bar.

$$dist = \frac{w(\frac{cot(\theta)}{d})}{2} \tag{3.3}$$

Where θ is the field of vision of the camera, *cot* is the *cotangent* function (that is, the reciprocal of the tangent) of the camera's field of vision, d is the normalised distance in image space of the light balls, and w is the width of the light sensor in centimetres. Figure 3.3 demonstrates this with an example field of vision of 30°.

This distance calculation does however become progressively more inaccurate the further off centre the user is, as the effects of perspective will bring the light blobs closer together when seen from the Wiimote camera. Figure 3.4 demonstrates this: despite being an equal distance away from the sensor bar in each case, Wii Remotes a and b see different images, with the viewpoint of b being such that the light sources appear closer, leading to an incorrectly calculated distance.

Assuming the user is standing such that they can point the Wii Remote directly at the sensor bar, it is also possible to determine the amount of rotation around the y axis (also known as *yaw*). Using the image space position of the IR sources, yaw can easily be deduced as an interpolation between 0 and 30°, shown in Figure 3.5. When used with the controller held forward, this affords determination of the rotation around the gravitational axis, which is not usually possible, as previously outlined in Section 3.3.2. Clearly, this yaw determination breaks down in cases where the IR lights cannot be seen, and the narrow field of vision of 30° limits the usefulness of the IR camera in a rehabilitative context, as the larger motions of the hand during therapy can quite easily move beyond the range in which the sensor bar can be seen [ALMK10a].

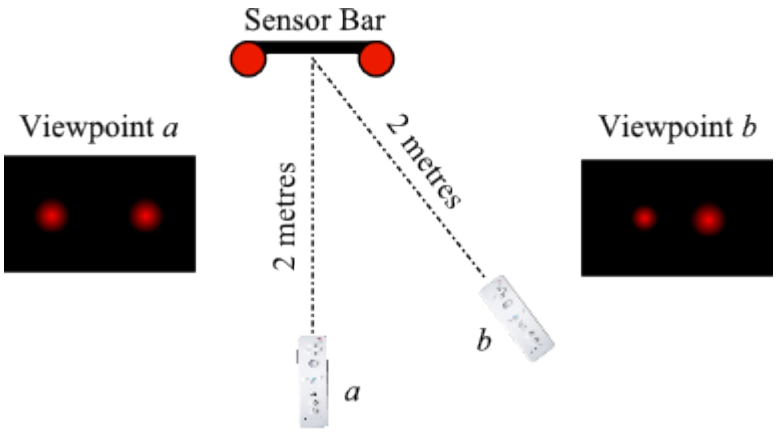


Figure 3.4: Limitations of distance calculations using the Wii Remote camera. If the Wiimote camera is viewing the LEDs at an angle, the relative positions of the lights can result in an innacurate position being calculated.

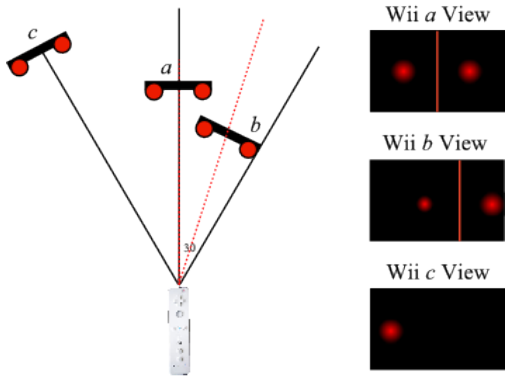


Figure 3.5: Limitations of yaw calculations using the Wii Remote camera. The camera-space positions of the sensor bar LEDs can be used to calculate some degree of yaw when the Wiimote is pointed forwards, however the limited field of vision of the camera can lead to loss of LED tracking, reducing yaw accuracy.

Figure 3.6: Right: The PSEye camera. Left: The PS Move wand. The large coloured orb of the PS Move wand allows image tracking algoithms to be used for accurate positional tracking.

3.3.4 PlayStation Move

The Sony *PlayStation Move* device was released for use with their *PlayStation 3* console, and takes the form of a hand held 'wand', similar to the Wii Remote. Rather than the IR camera of the Wii Remote, the front of the Move has a semi-transparent plastic ball approximately 2 inches in diameter, that can be illuminated by programmable RGB LEDs inside the controller. Like the Wii, the move device contains a 3-axis accelerometer, and a 3-axis gyroscope as in the Wii MotionPlus. In addition to these sensors, the PS Move also contains a *magnetometer*, which measures magnetic fields. This can be used as a compass, affording a fixed frame of reference to orientation tracking algorithms.

The PS Move controller comes as a package with the *PlayStation Eye* camera. The camera is capable of detecting a full RGB image at a resolution of 640x480, with a refresh rate of up to 100Hz. As with the Wii Sensor Bar, the PS Eye is designed to be placed above or below the player's television, where it can be used to record the player's movements, and audio using its inbuilt array microphone.

While the magnetometer affords greater orientation detection accuracy, the sensors onboard the PS Move are insufficient to produce an accurate 3D position over time, for the same reasons as the Wii discussed in Section 3.3.2. Instead, the 3D position of the PS Move is derived utilising the camera. The image space position of the glowing ball on the end of the PS Move device can be found by image processing, and then a 3D position calculated as described in Section 3.3.3, utilising the radius of the lit circle in the image, rather than the distance between Wii sensor bar LEDs. As the wand ball is sufficiently bright the PS-Eye camera can be operated with low exposure to eliminate most of the light reaching it without losing the track of the ball, affording easier detection within the camera image, with results easily verified against the known colour that the wand ball has been set to. This hardware combination allows for computationally inexpensive detection of the PS Move wand's

Figure 3.7: Top Left: The Kinect For XBox 360 device. Bottom Left: The Kinect for XBox One. Right: Kinect skeletal tracking joints, determined using a machine learning algorithm within the Kinect software from the colour and depth images. Images from microsoft.com

position in relation to the PS Eye camera, compared to algorithms designed to track shapes within a live video feed. Accurate orientation can then be derived from the gyroscopic and accelerometer sensors.

Sony have not released an official method by which to directly connect a PS Move or PS eye to a PC, and while third party drivers and libraries exist to facilitate this, the work outlined within this chapter instead utilises Sony's *MoveMe* software (store.playstation.com; *Sony Computer Entertainment*); this runs on a PlayStation 3 console, and allows other devices to connect via TCP/IP, and receive a live feed of the PS Eye camera, as well as the position and orientation of any Move devices being tracked, at a rate of up to 60hz.

3.3.5 Kinect

Microsoft's *Kinect* motion sensing system consists of a RGB camera, an infra-red laser pattern projector and an infra-red camera, packaged together in a motorised bar (shown in Figure 3.7) to be positioned above or below the user's television, as with the Wii sensor bar and PS Eye. The device utilises these features to monitor the actions of the user directly, so no additional hand-held devices are required. While the exact method of depth determination has not been made public knowledge, it is known that the Kinect features an infra-red projector, emitting a fixed random array of points, as shown in Figure 3.8. Objects in the field of emittance of the infra-red projector will cause distortions in the position, shape, and size of the dots from the IR camera's viewpoint; this allows for distance from the Kinect to be calculated, as the greater the distortion, the closer the occluding object must be. Depth is reported by the Kinect software at an 11-bit resolution, allowing for 2048 unique values. It has been determined that these values are not linear [KE12], with greater depth detection accuracy the closer the occluding object is to the camera sensor. As depth

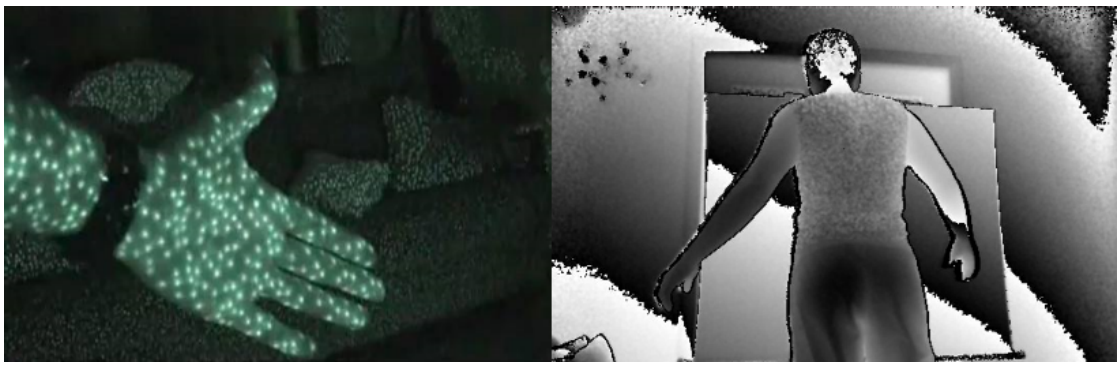


Figure 3.8: Left: Infrared projection of dot pattern. Right: Depth Image produced by Kinect. Left Image taken from graphics.stanford.edu/˜mdfisher/Kinect.html, Right image taken from pterneas.com/2014/02/08/kinect-for-windows-version-2-overview/

is determined by translation of a fixed number of projected infra-red dots, the device is not capable of a 'true' depth value per camera pixel - instead depth is derived via interpolation between the determined depths of the visible points.

Utilising the depth image, and machine learning algorithms trained over thousands of test images, the Kinect software provides developers with a 20-joint skeleton approximating the tracked person's limb pose, updated at a rate of up to 30hz. Joint information is provided as a hierarchical model, with joints having a parent/child relationship, and a position and orientation relative to that of their parent. Microsoft provide an official software development kit for accessing this hierarchical model, which utilises callbacks to inform the host program of new pose information becoming available; this software was used for the work described in this chapter. Joint position and orientation accuracy is graded by the SDK, allowing for the program to ignore data the SDK deems unlikely to be accurate.

3.3.6 Kinect 2

Late 2013 saw the release of an improved Kinect model, for use with Microsoft's new flagship *Xbox One* console. An incremental improvement over the otherwise similar Kinect, primary improvements are an improvement in the field of vision, allowing capture of user motion from 3 feet away, rather than the 6 feet of the original device. Additional joints have been added to the skeletal model, allowing for thumb and limited hand shape tracking. Basic facial expression detection, and heart rate monitoring via infrared have also been added to the device.

The colour data has been improved to a resolution of 1080 x 1920, and the depth resolution changed to a 'true' per-pixel resolution of 512 x 424, with the depth sensor changing from the random dot projection method of the original Kinect, to a 'time-of-flight' camera, in which the time taken for a strobed laser to hit a surface and reflect into the high update frequency depth sensor is recorded, using the known

Figure 3.9: Left: The Sixense Truemotion. Right: Razer Hydra. Both devices utilise the Sixense magnetic technology to produce positional and orientation information. Left image from www.cgsociety.org, right image from www.razerzone.com

speed of light to calculate distance of the reflecting surface from the sensor [kin].

As with the first generation Kinect, Microsoft provide an extensive SDK with which to develop programs utilising the improved Kinect's features, and this was used to interface with the device for the work in this chapter.

3.3.7 Sixense

The *Sixense* motion control system from Sixense Entertainment Inc. uses magnetic motion tracking to provide continuous position and orientation information. The use of electromagnetic fields is well established technology for reliably measuring three-dimensional space [Han87]. The Sixense comprises of a base unit connected to a host pc via USB, and up to 4 wireless hand held controllers. The base unit emits a rapidly switching magnetic field, that can be detected by 3 orthogonally placed magnetometers within each of the controllers, providing the reference for the position and orientation. The use of magnetic field detection means that unlike the other motion detecting devices discussed in this chapter, the Sixense is not hampered by camera line of sight issues, but tracking accuracy may be affected by metallic objects distorting the magnetic field. The controllers connect to the base station via a proprietary wireless connection, so the host application only communicates with the base unit itself.

The Circus Challenge game outlined in Chapter 2 utilises 3 Sixense controllers connected to one base unit; one in each hand, and one tucked into the belt or pocket at waist level to provide further data on the patient's base position and motion. By utilising the relative position of the hand held controllers to the waist controller, controller movements can be differentiated between arm movements and whole body movements, affording the user some degree of freedom to move around the Sixense detection area without impacting the accuracy of hand tracking.

The base unit was located at shoulder height in front of the user such that it does not block the game screen, but is within the centre of the full range of vertical upper arm freedom of movement. The distance between the patient and the base unit is also important, as magnetic field measurement is known to decay in strength and to distort as the distance between the source and sensor increases [Zac97].

3.4 Experiment Overview

In order to test the overall quality of the chosen motion detection devices, a test procedure was devised that would take into account the device's ability to accurately track position and orientation in all axes of 3D space, a requirement in order to fully replicate the 'real world' facsimile movements commonly employed in stroke recovery measurements. This test required a number of participants to use each of the tested devices to perform a number of simple movements, each of which was designed to reveal some aspect of the device's ability to accurately track movement.

To the author's best knowledge, there is no existing standard methodology for testing the accuracy of motion controlling devices, either within research literature or as part of the marketing or technical material of the devices tested. This is partially due to the differing technologies and use case scenarios of the devices - the Sixense is used to replicate hand motion in-game, while the Kinect is designed around whole-body tracking, for example. Many of the sensors embedded within the devices have datasheets available that *do* state minimum or maximum detected ranges (the Wii Mote accelerometer is an Analog Devices ADXL 330, with a datasheet stated specification of $\pm$ 3.0g with an error margin of 10% [Dev]), but as their output may be augmented by other sensors, or filtered in some manner before forming the final output of the device, it is not useful to look solely at such values in isolation when considering motion device overall accuracy.

To accommodate the differing technologies and use cases of the devices, a testing procedure had to be devised that would accurately determine both orientation and positional accuracy, while not favouring any one device over another. The test procedure must also be simple to perform and be understood by experiment participants, to allow data to be easily generated, and also easy to determine the results of computationally - as little filtering or processing of the signal should be performed so as to avoid adversely affecting the validity of the data being processed.

To this end, a simple experiment based around detection of circular motion was devised, whereby participants are requested to move their arm in a circle while

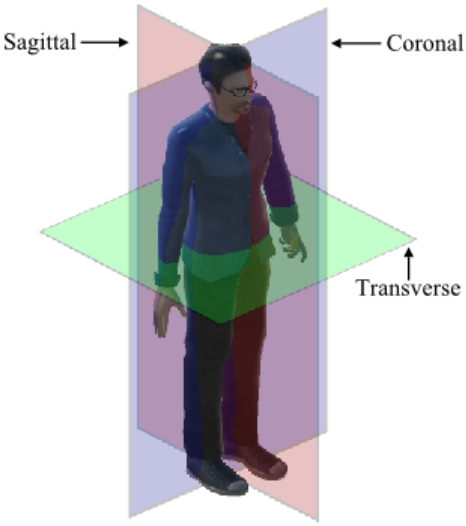


Figure 3.10: The three anatomical body planes: Sagittal, coronal, and transverse.

holding the sensing devices (where applicable); this allows for an intuitive test procedure, and a simple computation of the device's ability to track correctly. This test is performed in a number of different axis, as therapeutic motions may involve a variety of motions in 3D space around the user. By testing multiple directions, at a number of different distances, all devices will be treated the same regardless of their hardware features, and any particular strengths or weaknesses of the tested devices discovered.

3.4.1 Methodology

The test takes the form of a simple rotation in all three body axes - the *sagittal*, *coronal*, and *transverse* planes (as in Figure 3.10). The arm is held outward, holding the tracking device outward where necessary. The arm is then rotated 360° in the case of the sagittal and coronal plane, while the transverse plane is completed by having the test subject rotate around on the spot using their feet. By testing in all axes, a variety of test conditions are encountered. Accuracy in the coronal plane is primarily due to standard image tracking algorithms, while high quality data in the sagittal plane requires accuracy in determining a z axis movement change. The transverse plane provides a measure of a device's ability to recover from 'hard error' in that the user's own body will obscure the their hand during rotation, along with anything held within it. Poor test results in this plane compared to the sagittal plane are likely indicators that the device is losing accurate tracking, and cannot quickly recover tracking once the user has rotated enough such that they no longer obscure their hand.

All of the devices to be tested are likely to degrade in tracking ability as the user moves away from their 'base unit'; either as a product of camera resolution

limitations, or weakening of magnetic field influence. To determine the level of this degradation, all planes are tested at a distance of 1, 2, 3, and 4 metres. The camera or base station of the tested device is placed at a height of 150cm from the floor, and oriented such that it is facing directly forwards, in line with the metre markers participants use to position themselves.

Test participants were chosen to depict a range of body dimensions, ranging from a female of 5'2", to a male of 6'5". All of the participants were recorded at all 4 distances, using the Kinect, Kinect 2, Sixense, and PS Move, in a total of 16 separate runs.

So as to reduce variability in test results, a strict set of instructions are provided to test participants at the beginning of each of their recording sessions, to be enacted in order:

1. Stand with heels on correct metre marker
2. Place hands by side
3. Move hand in direction indicated at comfortable speed
4. Stop when 3 full rotations have been completed

The user completes 3 full rotations per run, with only data recorded from the second rotation counting towards the final result; this reduces the influence of participant error when beginning or ending their movement. A rest period is allowed between each plane and distance switch, to reduce arm strain, and potential for dizziness, particularly after completion of the transverse plane test, as it involves full body rotation.

3.4.2 Metrics

Each of the devices investigated in this chapter are capable of providing both 3D positional and orientation tracking, through a variety of means. In order to determine the overall quality of this tracking, it is necessary to determine metrics by which to measure them. As the focus of this work is the use of video game technology for stroke rehabilitation of the upper limb, the most relevant tracking point common to all devices is the user's hands, and so only this tracking is considered, even in devices capable of more complex position detection.

3.4.3 Position Accuracy

To produce a measurement of positional accuracy, only the relevant 2 axes for each plane from each trial-run were considered, creating a sequence of 2D coordinates

that can be considered as the vertices of a 2D polygon. Each trial's data is then centred around the 2D origin, according to the *centroid* of the polygon, calculated from all n positions p in the sequence s as:

$$Centroid(s) = \frac{p_1 + p_2 + \cdots + p_n}{n} \tag{3.4}$$

Once the captured data has been transformed in this way, it is possible to determine two metrics by which to rate the quality of the data: *position circularity*, and *orientation drift*.

The circularity of data set s can be defined using the following common shape factor:

$$Circularity(s) = \frac{4\pi a}{p^2} \tag{3.5}$$

Where a is the area of the shape, and p its perimeter. This circularity measurement results in a value within the range [0,1].

As we area dealing with 2D coordinates, the area of each polygon can be calculated using the Shoelace algorithm:

$$Area(s) = \frac{1}{2}|\sum_{i=1}^{n} p_i x p_{i+1 \oplus n} y - \sum_{i=1}^{n} p_i y p_{i+1 \oplus n} x| \tag{3.6}$$

Wheras the perimeter is defined as:

$$Perimeter(s) = \sum_{i=1}^{n} ||\overrightarrow{p_{i+1 \oplus n} - p_i}|| \tag{3.7}$$

The circularity factor is used as a metric of *positional* accuracy; deviations in positional tracking will result in changes in area and increase in perimeter length, lowering the circularity score. Deviations in position orthogonal to a tested plane will not be accounted for, but this will be compensated for by tests in the other planes.

3.4.4 Orientation Accuracy

A measure of accuracy of *orientation* can be formed in a similar way to that of position. By transforming the relevant forward basis vector for the input device by the orientation calculated from the recorded sequence, a normalised vector pointing

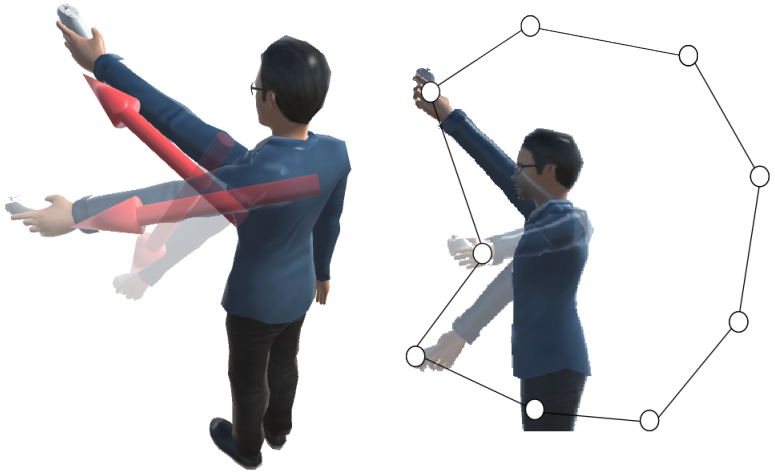


Figure 3.11: Left: Orientation mapped as normals. Right: Projecting normals onto sagittal plane results in a 2D polygon, the circularity of which can be used to measure arm orientation accuracy.

in the direction of the device can be created. By projecting this vector onto the plane being tested, a 2D position can be formed. For directions that lie exactly on the plane, the point will lie at a distance of 1.0. Deviations in the direction away from the plane will result in projected points closer to the origin. The projected direction vectors over the course of one movement will form a 2D polygon, as shown in Figure 3.11, that can be tested using the same circularity measurement discussed previously. This method will not take into account deviations in rotation *around* the test axis, but tendencies towards inaccuracy in all axes will be discovered by testing of all three body planes.

3.4.5 Overall Quality

As the circularity calculation provides values in the range [0,1], the circularity of position and orientation can be used to calculate a measure of data quality as a simple arithmetic mean:

$$Quality(s) = \frac{Circularity(s_p) + Circularity(s_v)}{2} \tag{3.8}$$

Where p and v are the position and calculated forward vector data sets of test sequence s.

3.5 Data Recording and Analysis

In order to facilitate the recording, visualisation, and analysis of the data obtained from the test participants, a number of programs were created. All were programmed in the C++ language using Microsoft Visual Studio (www.visualstudio.com/;

Microsoft Inc), and utilised the *Qt* toolkit (www.qt.io/; *The Qt Company*) for their graphical user interface, with OpenGL (www.opengl.org; *Khronos Group*) used for hardware accelerated 3D rendering where necessary.

The first of these programs was used to record participant data, and thus was linked against all necessary libraries to communicate with any of the tested devices. Upon selecting a device and successfully connecting to it, a window displays the position of a tracked device, superimposed over a live video feed if available. This allows for correct calibration of the device, ensuring that it is placed correctly to successfully see the test participant. On screen vocal delivery cues are presented to deliver to the test participant, and upon pressing the play button, data is recorded to a file, labelled automatically with the test participant's name, the current body plane, and timestamp. Data is saved to file as frequently as the motion device allows, and consists of a constant stream of 3D positions, 4D orientations, and time since recording began. All values are floating point, with positions being native measurements of the device, and orientations encoded as unit length quaternions.

Previously recorded data can be loaded into a second program, which allows for visualisation of a dataset in 3D space, rendered on screen using the *OpenGL* graphics library. In addition to visualisation, it allows for cutting of datasets. With a data recording loaded, it is necessary to crop data to only the second rotation based on selecting beginning and ending timestamps, determined from a playback 'scrubber' bar.

The beginning timestamp was always manually selected using the scrubber bar, as it was found that automatic methods were highly error prone - due to inaccurate sampling in some devices, particularly at distance and in the transverse plane led to high error rates in automatic detection attempts, also it was found that some test subjects would not necessarily strictly follow instructions - swinging their arms to 'warm up' before beginning their rotations, or switching the arm used to hold the controller between play sessions.

An attempt at finding the end of a dataset *can* be attempted by the program, by automatically progressing through the dataset to find the next nearest data point to the start, after having progressed through the dataset to pass through a point directly opposite the start, as demonstrated in Figure 3.12, in which the nearest point to starting point a is actually incorrect due to a noisy signal (point b), and issue which can be alleviated by seeking for the nearest data point only after point c has been traversed. This simple rule helped reduce occasions where inaccurate position sampling by the device (or user error) caused the movement to pass back from the

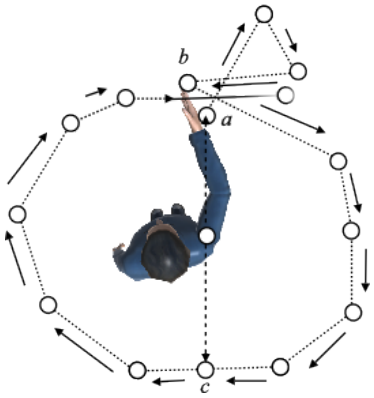


Figure 3.12: Automatic inference of a rotation end, utilising the direction of travel, and determination of nearby waypoints. While position *b* is closest to starting point *a*, it is actually the result of noisy data, and should not be considered the end waypoint

starting point. Manual inspection of the selected data point is performed, as this method could still sometimes fail, but generally saved time during the process of cropping the data.

A final analysis program is ran to programmatically generate comma separated value files, and generate plots for each dataset, using the *QCustomPlot* plugin for the Qt toolkit (www.qcustomplot.com; *Emanuel Eichhammer*). This program takes in a set of all *n* samples from a previously recorded file, and produces quality metrics from the position and orientation data, and generates x-y plots of the position and orientation data. This process is automatically undertaken for every dataset in a user-specified source folder, including any subfolders, allowing for a large amount of data to be generated with a single user operation. Upon completion of individual dataset analysis, the program generates per-device, and per-distance quality metrics, derived from the mean of all relevant datasets.

3.6 Results

3.6.1 Considerations

It is worth noting that the recording methodology accentuates weaknesses with certain types of motion detecting device. Of the devices chosen, The Kinect and Kinect V2 are the only ones which are entirely camera based. This results in cases of 'hard error', due to the lack of information when the camera cannot see the user's hand. The MoveMe uses camera tracking for positional data, so it too will suffer from a drop-out in positional information when the Move 'wand' cannot be seen by the PS Eye camera; although orientation can still be tracked from the sensors contained within the Move device, for the purposes of this experiment they were ignored when the position was not determinable, to preserve the 1:1 mapping of position and orientation samples.

Device	Sum Distance	Sum Angle	Distance Travelled	Angle Travelled
KinectV1	0.084	0.6938	0.0037	0.00962
KinectV2	0.955	149.878	0.00091	0.7196
MoveMe	2101.93	8.835	8.9441	0.00169
Sixense	227.936	1.2120	4.2719	0.00826
MotionPlus	4281.13	2.254	4272.98	0.20264

Table 3.1: Device Steadiness Comparison: Travelled columns represent delta from first measurement to last, with sum representing the accumulation of delta from measurement to measurement. Angle measured in degrees. Distance measured in cm.

This weakness is expected to show itself primarily when recording movements in the transverse plane, as the user's body will obscure the camera's view of the hand / device, with some additional obscuration expected with the sagittal plane recordings when the hand is facing away from the camera. While any loss of data is obviously undesirable, the ability for the devices to re-detect and track the desired movement after obscuration is a useful metric when considering the efficacy of motion devices for rehabilitative purposes.

It must also be noted that a device does not necessarily have to be capable of a final Quality result of 1.0 to be considered to have 'perfect' tracking. As the user is required to move their arm in a circular pattern, the maximum Quality that can be generated is limited by the movement of the test subject's shoulder joints, and their ability to keep entirely to movement across the tested plane. All test subjects self-report as being fully able bodied, and stated that they were comfortable with the motions required; however results of movement in the sagittal plane was found to have a slight outward movement towards the top of the movement in most users as they bring their arm around. While each test participant was of a different stature, the use of normalised directions and positions in the Quality metrics means that any such movement deviations impact each user's score by the same amount, allowing comparisons to still be made.

3.6.2 Data Output At Rest

To provide a baseline measurement of expected quality from each of the devices, a 30 second recording of the device at rest was taken; in the case of the Kinect devices, this required a volunteer to sit still, with their arms being tracked. The recordings were taken at a distance or two metres, as this was the minimum distance covered by all of the tested devices. The results of this initial test are tabulated in Table 3.1. This shows the sum of the changes in distance and orientation angle between the samples in a 30 second recording of the device at rest, and the total change in position and orientation between the first and last samples.

Immediately obvious is the poor result in movement from the Wii RemotePlus - the sum distance is several orders of magnitude larger than any other, due to the accumulated error in the double integration of position from acceleration. Both the MoveMe and Sixense also have large sum distances recorded, but, due to the low change in final position, it can be inferred that this was due to a high frequency 'jitter', rather than a large drift over time. As the Wii RemotePlus will report an acceleration value without any inbuilt processing to counteract gravity, this was accounted for by taking the mean of a short sample of the RemotePlus accelerometer vector at rest, and negating the result, transformed by the orientation of the device. Evidently this was not sufficient to reliably remove gravity from the device reading, and further processing would be required to create a stable result.

The Kinect2 shows a large accumulated sum angle, despite a low distance travelled. Due to the method in which orientation is determined from the skeleton calculated by the Kinect software this is a curious result, as it suggests that the 'parent' joint of the skeletal hierarchy must have had a large drift or jitter in its position. Visual inspection of the entire skeleton provided by the Kinect software does tally with this result, as the pose presented tends to distort frequently. The low distances for the Kinect devices are at odds with the other tested devices, suggesting additional filtering of position, while the low changes in angle over time as compared to position for all devices is a good indicator as to the relative difficulties in determining these values.

3.6.3 Sample Rates

The devices tested communicate with their host PC in a variety of ways - The Kinect via direct USB connection, the Sixense wirelessly to a USB connected base unit, and the PS Move wirelessly via Bluetooth to a PlayStation 3 running the MoveMe server, which then sends the results to the host computer via TCP/IP. Due to these connection differences, and the differing ways in which the controllers derive position and orientation information, it is useful to compare their respective update rates, as reported from the interface software.

Device	Coronal	Transverse	Sagittal
KinectV1	70.125	117.0	56.625
KinectV2	36.381	75.864	34.9
MoveMe	140.583	220.231	144.333
Sixense	137.667	188.267	142.933

Table 3.2: All devices Sample Count Chart: Mean sample count of each body plane across all recordings at all distances

The mean sample counts for each device across all recorded trial runs are collated in Table 3.2. It can be immediately noted that the Kinect devices update at a significantly lower rate than the MoveMe and Sixense devices, which both report at a similar rate. The effects of sample rate upon the resulting quality of movement detection is discussed in the Kinect section. It was found after test completion that the transverse plane took on average longer to complete than the other planes ($\approx$ 2.5 seconds, compared to $\approx$ 1.35 and $\approx$ 1.45). Adjusting sample counts based on time taken results in Table 3.3. As expected, this reveals the lower sample count per second in the transverse plane for both the MoveMe and Sixense controllers, due to camera obscuration and greater distance from the base unit respectively; however, the Kinect devices tended towards higher. This can perhaps be explained by the tendency of both Kinects to generate highly inaccurate skeletal tracking poses when limbs are obscured from the camera, resulting in the high angle travelled noted in Section 3.6.2.

Device	**Coronal**	**Transverse**	**Sagittal**
KinectV1	51.5523	46.448	38.7132
KinectV2	26.7454	30.1178	23.8603
MoveMe	103.3493	87.431	98.6771
Sixense	101.2056	74.7415	97.7199
Mean Time	1.3603	2.5189	1.4627

Table 3.3: Sample Counts adjusted for time: Different body planes took differing amounts of time to complete on average. Dividing mean sample counts of Table 3.2 by time allows a measurement of samples per second for each device. Mean time measured in seconds.

3.6.4 Overview Of Circularity Results

The circularity metric calculations for each device are shown in Table 3.4 and Figure 3.13. Dashes indicate distances for which no data could be recorded for that particular device, due to insufficient distance between the test participant and the detection device for it to correctly detect all of their movement. To aid in the visualisation of these results, they are additionally collated graphically in Figure 3.13.

3.6.5 Wii MotionPlus Results

The position of the Wii MotionPlus device can be determined in one of two ways: Either via double integration of its accelerometer data over time, or by processing the position of the IR light sources from the tracker bar, as discussed in Section 3.3.1. Unfortunately, the experiment as designed is unsuitable for position determination via the tracker bar, as the bar will never be in the line of sight when rotating in

Device	Distance	Orientation	Position	Quality
KinectV1	1	-	-	-
KinectV1	2	0.7048	0.6035	0.6542
KinectV1	3	0.5124	0.5958	0.5541
KinectV1	4	-	-	-
KinectV2	1	0.5294	0.5283	0.5289
KinectV2	2	0.6558	0.6849	0.6703
KinectV2	3	0.5392	0.5555	0.5474
KinectV2	4	0.5181	0.4888	0.5034
MoveMe	1	-	-	-
MoveMe	2	0.9752	0.6458	0.8105
MoveMe	3	0.9409	0.3863	0.6636
MoveMe	4	0.7634	0.2417	0.5025
Sixense	1	0.9805	0.9487	0.9646
Sixense	2	0.8749	0.8354	0.8552
Sixense	3	0.4476	0.4153	0.4315
Sixense	4	0.4967	0.3017	0.3992

Table 3.4: Device Circularity Comparison: Results of circularity calculation outlined in section 3.4.3. Results closer to 1.0 are better. Distance measured in metres.

the coronal plane, and only briefly when rotating in the transverse or sagittal planes. Accurate derivation of position from accelerometer data is notoriously difficult, as any noise or bias in the accelerometer data will quickly accumulate into large errors in position. As gravity will be detected by the accelerometers, it must be accounted for, and any inaccuracy in this calculation will again result in large errors in position. This process too is error-prone, as evidenced by the large values reported by the Wii in Table 3.1.

Despite the poor positional output, a test trial of rotating in the transverse plane shows excellent stability, as shown in a plot in Figure 3.14, of the x and z components of the transformed forward vectors of each frame. As positional data could not be reliably calculated, no further tests were performed using the Wii MotionPlus device.

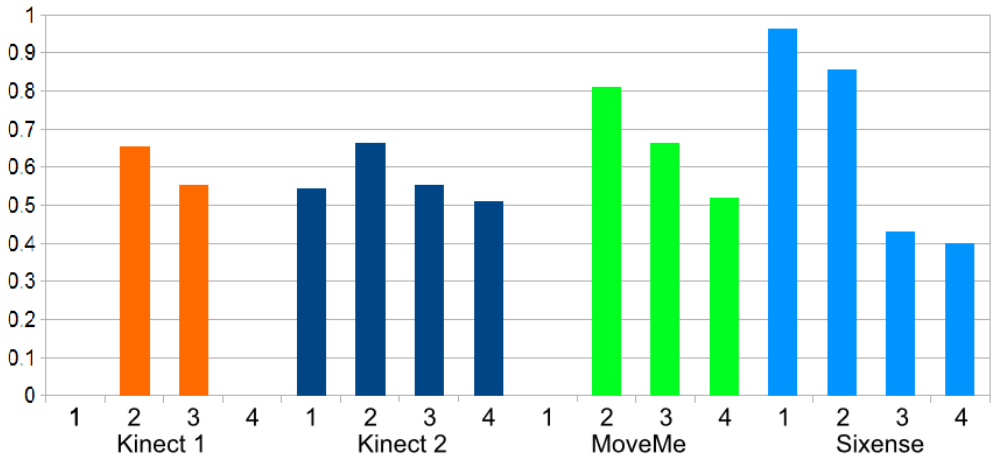


Figure 3.13: Device Circularity Comparison Chart, showing the results of the orientation quality calculation at a distance of 1,2,3 & 4 metres. The 1 and 4 metre results of the Kinect 1 are 0, due to being outside of the device's capable tracking range.

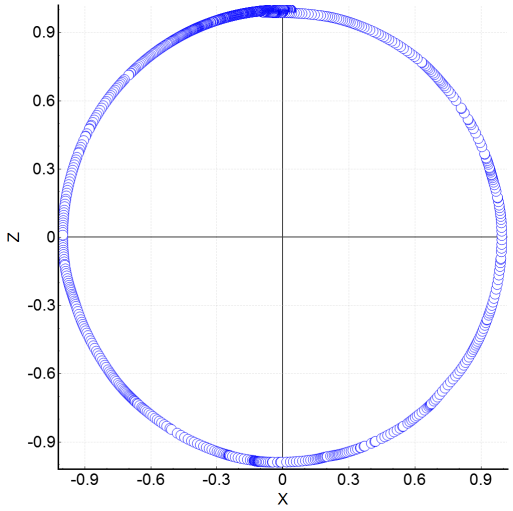


Figure 3.14: Wii orientation accuracy: The device is capable of a steady, high quality output rate in the transverse plane, with all samples being at a normalised distance of > 0.9.

3.6.6 Playstation Move Results

When trialling the MoveMe server setup, the 1 meter point was unable to be successfully recorded, as the field of view of the camera was insufficient to fully capture even the smallest trial participant. Recorded orientation information was considered excellent, with orientation circularity of ≈ 0.95 in all 3 planes at 2 and 3 metres, dropping to 0.76 at 4 metres. Unlike the other devices, orientation is derived from gyroscopes and accelerometers contained within the PS Move device, rather than a camera image or magnetic field, and so is unaffected by distance, however a reduction in image position tracking leads to fewer orientation samples being taken, impacting the quality of both. Position circularity at 2 metres was considered good (circularity 0.646, stdev 0.178), but displays a rapid drop-off at distance, resulting

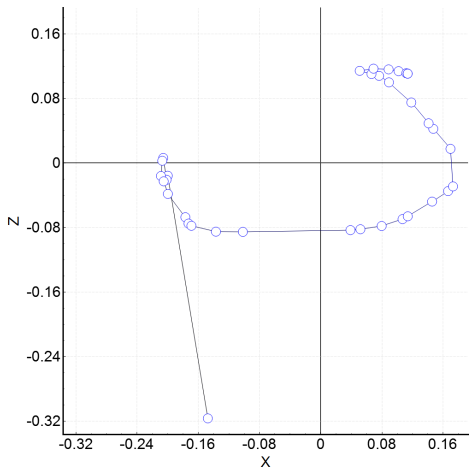


Figure 3.15: Plot of MoveMe at 4 metres in transverse plane

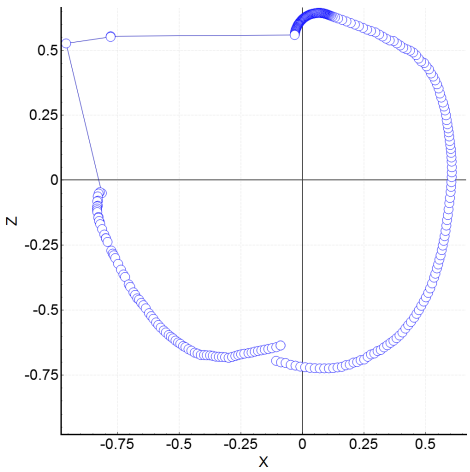


Figure 3.16: Plot of MoveMe at 2 metres in transverse plane

in a consistent poor result (circularity 0.38 stdev 0.024).

Fully accurate tracking of the MoveMe does start to degrade at 3 metres, with z axis tracking degradation prominent in several samples at 3 metres, and in most at 4 metres, but despite this movement in the coronal plane remained accurate. Figure 3.15 shows an example of a trial run from the MoveMe at 4 metres in the transverse plane, post-centralisation. Immediately obvious is the large gap in the top left quarter of the graph - the trial participant was rotating anticlockwise, with the beginning of the gap denoting the point at which the move became obscure by their arm, and the end the point at which tracking was regained, with an additional tracking fault in the bottom left quadrant. Data also becomes more ovular at distance, with an approximately 2:1 ratio between minimum and maximum point lengths in the plot axis (position circularity: 0.45). This can be compared to figure 3.16, in which a transverse plane recording at 2 metres shows an approximate 1:1 ratio (position circularity: 0.807).

The position detection limit of the MoveMe software appears to be approximately 4 meters, as it was noted in several trials that further movement in the z axis caused a 'cutoff' effect, where no further movement in the axis is seen, demonstrated in the right of Figure 3.17, a trial run which resulted in an orientation circularity of 0.81, yet only a position circularity 0.03.

3.6.7 Microsoft Kinect Results

Trials using both Kinect devices reveal that both behave similarly, and result in practically identical performance at 2 and 3 metres; however the Kinect 2 benefits from updated optics and processing that allow it to additionally track users at 1 and 4 metres. Figure 3.18 shows the similarity in the overall quality metric scores, with

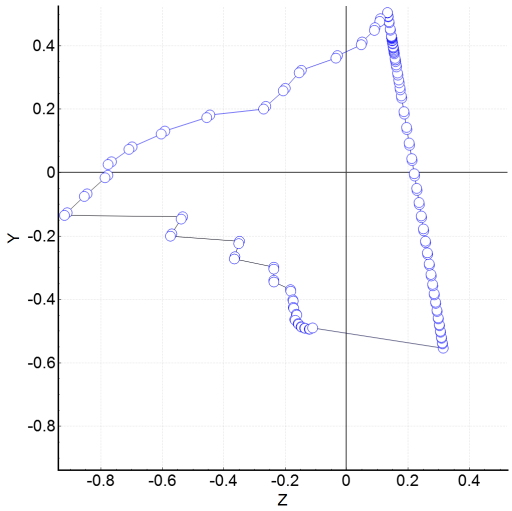


Figure 3.17: Plot of MoveMe showing cutoff effect at 4 metres along right of graph

the blue bars representing Kinect V2 quality results, and orange bars the original Kinect device.

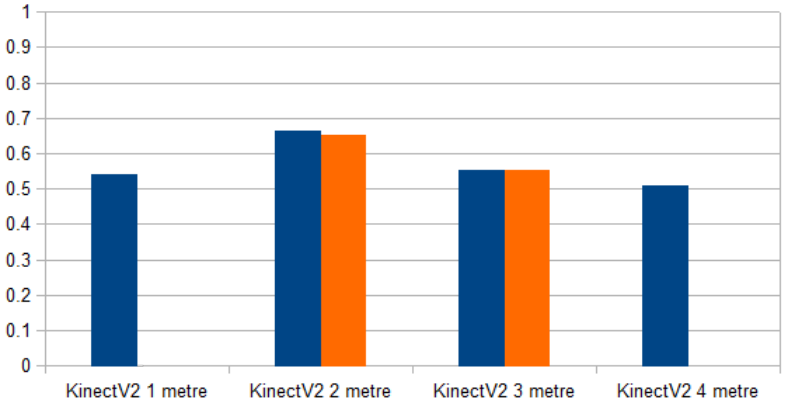


Figure 3.18: Kinect Quality at distance comparison. Results closer to 1.0 are more perfect in both orientation and position. Blue bars represent overall Kinect 2 quality at 1,2,3 &4 metres, while orange bars display the original Kinect qualities at the same distances.

As per Figure 3.4, the Kinect scores at 2 are noticeably lower than those of the other tested devices (0.65 for Kinect 1, 0.52 for Kinect 2, compared to 0.81 for MoveMe, and 0.85 for Sixense). By splitting up the quality metric on a per-axis basis in table 3.5, it can be seen where the weaknesses in the Kinect sensor lie:

Kinect	**Coronal**	**Transverse**	**Sagittal**
Kinect 1	0.839	0.204	0.807
Kinect 2	0.834	0.154	0.784

Table 3.5: Kinect Quality per axis: Results closer to 1.0 are better in both orientation and position.

Both Kinect devices produce a very poor mean performance in the transverse plane (Kinect 1 quality 0.204 Stdev 0.23, Kinect 2 quality 0.154 Stdev 0.143).

This is to be expected, as rotations on this plane will lead to cases of hard error, when the tracked hand becoming obscured for a period. However, comparison with the MoveMe in the transverse plane (mean quality 0.68, Stdev 0.288), a similarly camera-based solution, reveals particular weakness in this plane.

One explanation of this could be due to the method in which each device derives position. Both Kinect devices work by calculating an entire hierarchical skeleton of joints, with the position of the wrist bone relying on the rest of the arm being correctly tracked, thus requiring more of the arm to be visible than with the MoveMe, where the bright, uniquely coloured ball of the PS Move can be quickly found via simple computer vision techniques.

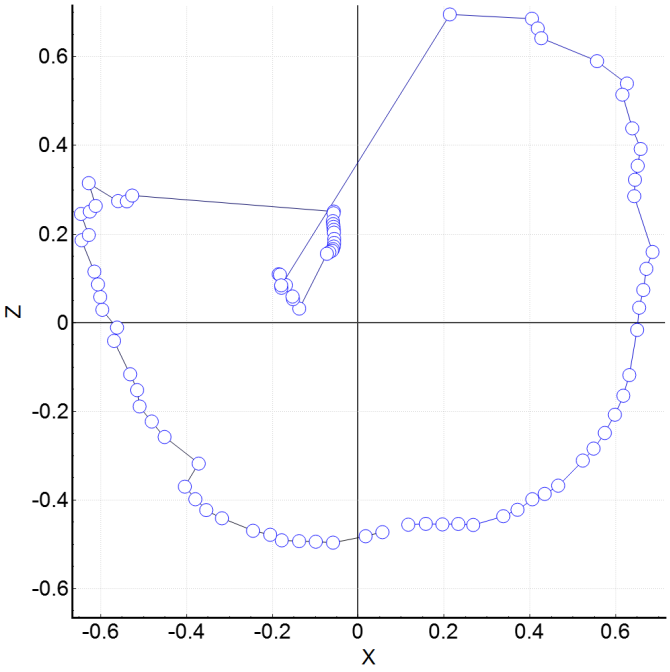


Figure 3.19: Plot of Kinect 2 at 2 metres in transverse plane. The large error in the upper left exemplifies how the kinect reacts to loss of tracking of the hand, resulting in significant deviations in positional tracking in each sample.

The quality scores of both Kinect devices are limited by the low sample rate of the devices, as shown in table 3.6. Lower sample counts will result in a more extreme 'squaring off' of the circular movement being recorded. An example Kinect 2 recording of the transverse plane at 2 metres is shown in figure 3.19. When compared to figure 3.16, the lower sample density becomes apparent. This data plot also shows a phenomena unique to the Kinect devices - as the person being recorded rotates such that their hand becomes obscured, the Kinect devices will attempt to seek out the hand elsewhere in the image, frequently picking up the left hand or other part of the body, thus the cluster of data points towards the centre of the plot. This is in contrast to the MoveMe, which will simply stop tracking until the ball is detected.

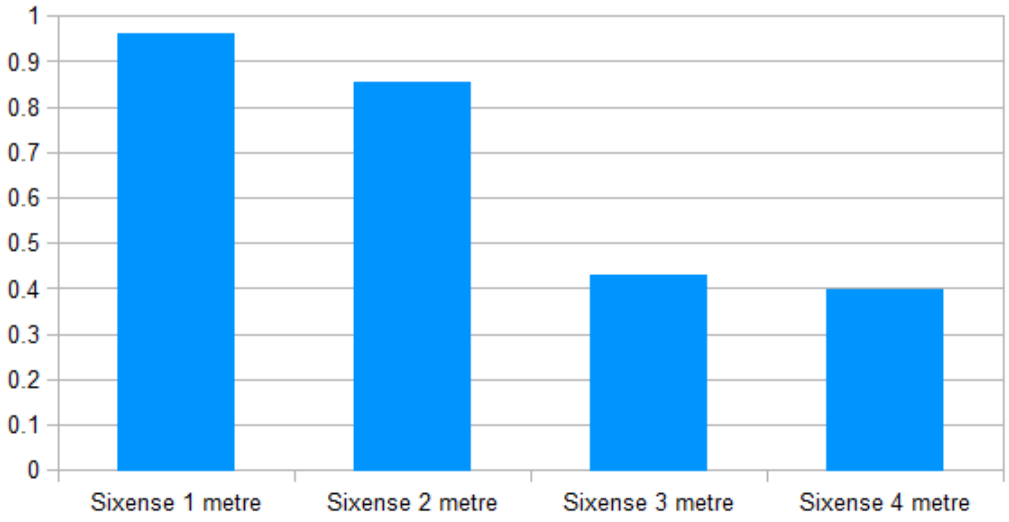


Figure 3.20: Sixense Quality Distance Comparison, showing the overall quality result at 1,2,3 & 4 metres.

Device	Axis	1 M	2 M	3 M	4 M
Kinect 1	Coronal	-	80.80	52.33	-
Kinect 1	Trans.	-	133.40	89.67	-
Kinect 1	Sagittal	-	62.40	47.00	-
Kinect 2	Coronal	34.80	35.56	40.17	29.00
Kinect 2	Trans.	69.83	79.00	77.00	78.00
Kinect 2	Sagittal	36.00	31.57	34.50	38.50

Table 3.6: Kinect sample count per axis: Distance measured in metres. Higher counts are likely to result in more accurate readings of true position and orientation.

3.6.8 Sixense TrueMotion Results

The Sixense is unique among the tested controllers in that it can accurately track position and orientation without the need for a camera. The performance of the device does degrade quickly with distance, however. This can be seen in a graph of its combined score metric vs. distance, in which a combined score of 0.96, drops to 0.43 and 0.39 by metres 3 and 4 (figure 3.20). This performance degradation is the worst of all devices over distance, and beyond 2 metres, the Sixense displays the worst quality metric of all devices. At a distance of 1 metre (or even less, as the device is not constrained by a camera's field of vision) however, the device is the best performer, with almost perfect circularity for both orientation and position, resulting in the excellent overall quality metrics shown in Table 3.4.

By examining the quality results for each axis and distance collated in Table 3.7, a general weakness in the sagittal plane can clearly be seen. Table 3.8 additionally shows the standard deviation of data samples for each axis and distance. Together, these show the high quality results of the Sixense at 1 metre, with particularly low deviation in quality observed in the transverse plane, a factor which separates the Sixense apart from the other tested devices. At 4 metres, the sagittal plane results in a consistently bad quality metric.

Axis	**1 Metre**	**2 Metre**	**3 Metre**	**4 Metre**
Coronal	0.9592	0.9521	0.4361	0.6648
Trans.	0.9847	0.9512	0.7023	0.3588
Sagittal	0.9497	0.6622	0.1560	0.1740

Table 3.7: Sixense Circularity axis comparison: Results closer to 1.0 are better.

Axis	**1 Metre**	**2 Metre**	**3 Metre**	**4 Metre**
Coronal	0.0257	0.0225	0.3634	0.3739
Trans.	0.0064	0.0243	0.1657	0.2119
Sagittal	0.0198	0.1837	0.2114	0.1075

Table 3.8: Sixense Quality deviation axis comparison across all tested distances. Numbers are standard deviation of overall quality of recordings made in each body axis

The extent to which the Sixense degrades becomes most apparent visually - Figure 3.21 shows the degradation in quality when taking recordings of movement in the sagittal plane, from an excellent results at a distance of 1 metre, to a complete breakdown of shape at 4 metres, making detection of circular gestures impossible.

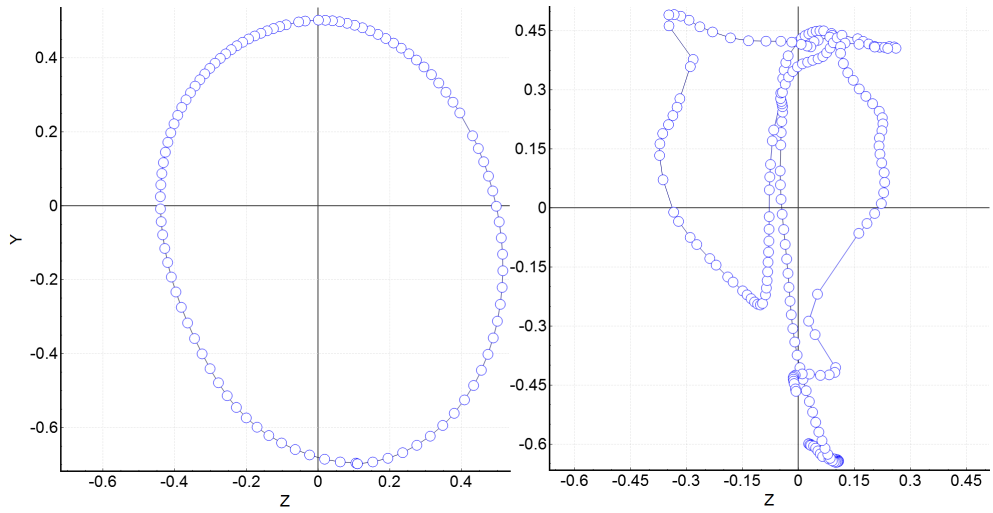


Figure 3.21: Left: Plot of Sixense positions at 1 metre in sagittal plane Right: Plot of Sixense positions at 4 metres in sagittal plane. Together, these display how much positional information degrades at distance using the Sixense device.

3.6.9 Discussion: Motion Sensing Devices

Devices designed to track hand movements are of great potential use within the field of rehabilitative gaming. However, in order to best utilise motion devices, it is important to factor in the relative strengths and weaknesses of each available device. The work described in this chapter enables an assessment of the ability of a number of motion devices to correctly detect the motions that could could make up a rehabilitative therapy.

The results obtained via this experiment serve as a useful indicator as to the quality of data pertinent to gesture recognition in the context of rehabilitative gaming. The most interesting outcome from this work is the excellent quality of data obtained from the Sixense device when used at distances close to its base unit, showing high accuracy in both position and orientation across usage in all 3 planes. This would suggest the Sixense would make an excellent device to use in situations where a high degree of fidelity in movement detection is required, such as detection of specific limb movements and poses. The device begins to heavily degrade at distance, however, so careful considerations must be made as to the environment it is used in - applications deployed via a laptop or tablet would be ideal conditions for the Sixense device.

Tests performed using the MoveMe setup show that the device produces quality positional and orientation information at 2 metres, with a degradation in performance beyond that. At 2 metres, all trial participants could be fully detected with limbs outstretched, making the MoveMe a good choice for living room interaction on a larger screen. The main downfall of the MoveMe server package is its reliance on additional external hardware - a PlayStation 3 console and a working network connection. Further work will be required to determine whether tighter control of the environment, in regards to any patches of light within the MoveMe's field of vision, would result in improved performance at range.

Although the Kinect devices score less well than the others, notably in the transverse plane, it should be pointed out that these devices provide data on the entire shape of the body, and in the Kinect 2's case, additional body metrics such as heart rate. When used at a distance of 2 meters, with no hand obscuration, these devices still produce quality data. At 2 meters, scores between the two devices are very similar, both displaying a weakness in recalculating its body tracking sufficiently to determine hand position during rotations in the transverse plane. As with the MoveMe, further testing is required to determine the sensitivity of the Kinect to differing lighting conditions; however, several trials with the Kinect 2 showed the device tracking the incorrect hand as the user turned away from the camera, sug-

gesting that processing of the camera image for facial detection to infer orientation could lead to a reduction in false readings.

3.7 Accuracy Metrics Of Force Sensing Devices

Many physical ailments can lead to a reduction in muscle strength in the hand and fingers [HWW+87] [KFCR06], and neurological conditions such as stroke can impair the use of a limb [HPBM01], resulting in muscle atrophy over time [SVA+13], causing further reduction of limb capability. Therefore, as part of the overall rehabilitation strategy of a patient suffering upper limb impairment, it is necessary to determine changes in ability to apply force with that limb; weaknesses in overall arm strength, or inability to apply force with precision lessens the ability to perform everyday tasks, reducing independence and quality of life. Many of the metrics used to gauge physical condition have sections that focus on upper limb dexterity and strength, such as a tests involving writing with a pen, or picking up and pouring a jug of water [TMC+]. To fully realize the concept of a complete home-based physical rehabilitation routine based on video game technologies, there must be methods by which to accurately measure patient strength, and gameplay features that promote increased upper limb usage.

For patients suffering acutely from the effects of stroke, it may be that only small limb movements are possible at all, making devices designed for detection of gross movement unsuitable. Therefore, other means by which a video game based rehabilitation therapy can be engaged with must be investigated. In such cases, it may be better to create gameplay mechanics around what limited ability a user has, again by measuring the ability to apply force, rather than the ability to move the entire upper limb.

In this section, a number of force sensing devices are examined, and tested for their accuracy. All of the devices discussed here were used as part of the patient trials of the rehabilitative software discussed in Chapter 4. There are relatively few commercial force sensing game controllers on the market, so two of the tested devices are instead designed for commercial purposes such as stock weighting.

3.7.1 Tested Devices

3.7.1.1 Wii Balance Board

The most popular force sensing gaming device currently on the market is Nintendo's *Wii Balance Board* [Nin], an inexpensive device designed to be stood upon, and used to interact with games that measure weight and the user's centre of gravity. As

outlined in Chapter 2, the device consists of 4 sensors arranged in a square, each capable of detecting up to 37.5kg of force imparted upon it. While the work of Clark et al [CBP+10] suggests that the Wii Balance Board will be unsuitable for upper arm rehabilitative tasks due to its high minimum detectable change, the device's ubiquity and affordability makes it a good baseline by which to compare the other devices. As with the Wii Remote, Nintendo have not released any official means by which to connect and communicate with the Wii Balance Board to a personal computer; however, as it uses the standard Bluetooth protocol for connectivity there are a number of open source libraries that can be used to extract information from it. For the purposes of these tests, the *WiiUse* library is used, as with the Wii Remote discussed earlier.

3.7.1.2 Saitek X-65F

The Saitek X-65F (www.saitek.com; *Mad Catz Interactive, Inc.*) is a high-end joystick designed to be used with flight simulator software, with a retail price of approximately £250. It has a number of characteristics that made it worth investigating as a force sensing tool. Firstly, it has an unusual design for a joystick in that it has no moving parts - unlike most joysticks which pivot and/or rotate, with the user's input determined via potentiometers or IR sensors, the X65F instead detects force imparted upon it via proprietary sensors in the base of the unit, with the joystick shaft itself remaining stationary. Secondly, as an extension of this force detection feature, the joystick software reports user input in *kilograms of force*, rather than an arbitrary scale. The software controlling the joystick enables a customisable kgf reporting range, allowing for an accurately calibrated result. Being designed for gameplay which involves small adjustments and fast reaction times, the joystick is presumed to have been designed to have a small minimum detectable change, and a high sample rate.

3.7.1.3 Pressure Profile Systems ConTacts C500 - DLP-IO8

The Contacts *C500* load sensor is a commercially available device, designed for medical and automotive uses (www.pressureprofile.com/capacitive-sensors; *Pressure Profile Systems*). It is a capacitive sensor, designed to measure the capacitance between two electrically charged surfaces built into it - as force is applied to the device, the surfaces are pushed closer together, and a measurable increase in capacitance occurs. Each sensor is rated to detect up to 10PSI of force, equivalent to approximately 4.53 kgf. The sensor itself is contained within a small fabric pad, connected to an electronics project box, containing the part of the device that turns the capacitive measurement into an analogue signal. Further hardware is required to convert this analogue signal into a digital reading suitable for computer use.

Figure 3.22: The Saitek X-65F device: the shaft of the device does not move, but instead is used to measure the amount of force the user is applying to the device.

For the purposes of this experiment, four such C500 load sensors were connected to a *DLP-IO8* acquisition board (www.dlpdesign.com/usb/io8.shtml; *DLP Design*) - a small unit capable of receiving analogue signals, converting them to digital values, and transferring them to PC via a USB connection working as a virtual serial port. The DLP device receives ASCII commands via an SSH connection, responding with the current voltage read on a particular pin. This voltage can be converted into a force reading, with higher voltages corresponding to higher loads. This relationship is non-linear, however, with reported values differing slightly between sensors. Each sensor is provided with a spreadsheet, containing the results of a calibrated load performed by the manufacturer, as an ordered list of tuples of weight and voltage. This data was used in the experiment to calculate an approximate force from the non-linear voltage recordings, interpolated via a cubic Hermite spline.

3.7.1.4 Loadstar iLoad Mini

The final device tested for its aplicability is the *iLoad Mini*, manufactured by LoadStar Sensors (www.loadstarsensors.com; *Loadstar Sensors*). The device is a small metal disc, designed to be mounted under a plate, to be used as highly accurate scales for medical and safety applications. Each sensor is stated by the manufacturer to be capable of a minimum detectable change of 1% its maximum rated load of 10.0 kgf, potentially making the devices ideal for upper limb therapy purposes. Similar to the C500 devices, the iLoad Mini is a capacitive force sensor. The devices are connected to a DQ-4000 unit, also manufactured by LoadStar Sensors, which is individually calibrated to turn the non-linear voltage readings into binary data; as with the DLP-IO8 device, the DQ-4000 is a USB device acting as a virtual serial port, with communication occurring via SSH.

Figure 3.23: Left: The PPS C500, attached to a DLP-IO8 analogue to USB converter. Right: The LoadStar iLoad Mini device. Both devices are capable of measuring force using 4 individual sensors, and reporting the result over USB.

3.7.2 Experiment Methodology

For each device, three experiments were performed: one to determine the sensitivity of the device, one to determine whether their reported values drift over time, and one to determine the accuracy of the reported values at a number of different weights, placed upon the devices to impart force upon them. The values reported are measured against values recorded by digital scales, stated to be accurate to within 0.01g. Not all of the devices supported the same range of detection, but where possible devices were compared like-for-like.

3.7.2.1 Accuracy Test Methodology

The differing force detection mechanisms of the devices necessitated slight differences in the methodology employed in testing each of the devices. For the LoadStar and PPS devices, the sensors were arranged in a square, and a flat surface placed on to of them to support the weights. For each tested weight, the sensor values at rest were first measured, and the tare weight recorded for later subtraction. Then, the weight was applied, and the resulting values were recorded as fast as the device would report them for a duration of not less than 90 seconds. The first 30 seconds of applied force was discarded to remove inaccurate readings resulting from instantaneous force, and the results of the 4 sensors summed to provide a mean weight.

The Saitek joystick required a different approach, as none of the surfaces of the joystick shaft were suitable for balancing weights on. As the joystick is designed to be attached to a desk, the base unit of the joystick comes complete with 4 threaded bolt holes; it was therefore possible to mount the joystick onto a bespoke setup that securely held the joystick at a 90°angle, such that its x axis pointed up and down, and the shaft of the joystick extended outwards. It was then possible to attach a platform that hangs underneath the joystick, within which weights can be placed. The platform was attached around the centre of the shaft of the joystick, to most closely match the point at which the average user would apply force to the joystick.

3.7.2.2 Noise & Bias Test Methodology

For each device, a second experiment was performed, whereby the sensors were left for a longer period of 5 and a half minutes, in order to determine how bias and noise affects a device over time. The device was loaded using a weight of 5 kilograms, to avoid any issues with values being internally clamped to 0 that may have arisen from an unladen device. The standard deviation, and sum of differences between each sample are determined from this data recording. A trend line was calculated for each sensor recording using lest squares analysis, to allow a measure of bias over time. As in the accuracy experiment, for each tested weight, the first 30 seconds of data was discarded to remove any instantaneous force from the readings, providing a 5 minute data sample.

3.7.2.3 Sensitivity Test Methodology

Each device was also tested for its sensitivity - that is, its ability to respond to minor differences in applied force. This was achieved by determining the minimum detectable change in reported value. As all of the devices will have some degree of noise in the reported value, the ability to detect small changes will be influenced by the outcome of the bias test. To determine that differing levels of force could be determined a test protocol was devised that would find the minimum level of force change that could be reliably detected from the output signal from the devices. This took the form of adding weights to the device at 30 second intervals, and performing analysis on the resulting data. This was repeated a number of times with different weight values, to determine within 15 grams the minimum change that could be reliably extracted from the signal. The output of these test runs was analysed within the MATLAB software package (uk.mathworks.com; *The MathWorks, Inc.*), using tools for handling piecewise constant signals. As the data would ideally take the form of a number of 'jumps' in the recorded value, as weights were added, the chosen method of signal analysis used was *Robust Jump Penalisation*, performed using MATLAB plugins from Little and Jones [LJ11], as this form of processing naturally preserves such signal shapes. As with the other tests, the sensitivity test was performed with a baseline load to prevent clamping issues, and the first 30 seconds of each run removed.

3.7.3 Results

3.7.3.1 Noise & Bias Experiment Results

The results of the bias experiment are shown in Figure 3.24. Several results are made immediately clear from the graph, most notably the amount of noise in the signal received from the Wii Balance board compared to the other devices, and the high level of accuracy of the iLoad device compared to any of the others. Table

3.9 collates further results from the bias test; it shows the iLoad and Saitek devices as having particularly low standard deviation in recorded value over time, and that the Wii Balance board suffers from several orders of magnitude more noise in its signal, measured by the derivative of the *y*-axis value over the course of the experiment; this is to such an extent that accurate measurements of small values may be impossible.

Only the PPS C500 displays a noticeable bias over time, and has a notable increase in the amount of signal noise over time - the shape of the graphed data is better described as that of a cone than a line. Such signal output may result in the device being unsuited to tasks involving a constant amount of force being applied over time, such as centre of balance therapies, or for detecting small changes over time, as user input will be masked behind the natural bias of the device.

Figure 3.24: Graph displaying device noise and bias over time of each device. Noise is represented as high frequency oscillations (note the Balance Board results in green), while bias is shown as a constant shift in measurement over time (note the positive bias of the PPS500 in orange).

3.7.3.2 Wii Balance Board

Previous work utilising the Wii Balance Board has suggested that the device would be unsuitable for accurate measurement of applied force [CBP+10], and the accuracy test has shown this to largely be the case. Table 3.10 collates the readings taken using the device, showing the large range of error percentage - at all weights the device under-reported the true weight applied, except for the large overestimation at

Device	Total Diff	Std Dev.	Best Fit Slope
Wii Balance Board	210640.37	66.227	-0.078
Saitek X65F	2370.00	2.484	-0.008
PPS C500	22695.78	62.860	0.716
iLoad Mini	7172.65	3.258	0.014

Table 3.9: Bias Results for all devices: Diff is sum of differences from true applied force across all samples (in g). Best fit slope is the slope of a line of best fit of all data samples.

20g, which can be assumed to be due to the large amount of noise present in the device readings, as shown in the noise and bias test. The device does however improve in accuracy as more weight is applied - as the device is originally designed to be stood upon, it can be assumed that the onboard sensors are designed to be most accurate at the ranges around that of a typical person's weight. The device would therefore be unsuitable for realtime determination of small input forces, but could still prove to be useful in therapies closer in design to its original intention, that of games based around a shift in the centre of gravity, possibly as part of a therapeutic routine for balance improvement and core muscle strength.

Despite the level of noise in the signal, the sensitivity test resulted in an accurate determination of 5 50g weights applied to the device (Figure 3.25 Right), failing at 35g by not detecting the first applied weight (Figure 3.25 Left). This suggests that the Wii Balance Board would be useful in cases where it is only necessary to monitor force readings to determine compliance with a force-based routine, rather than detecting accuracy of applied force.

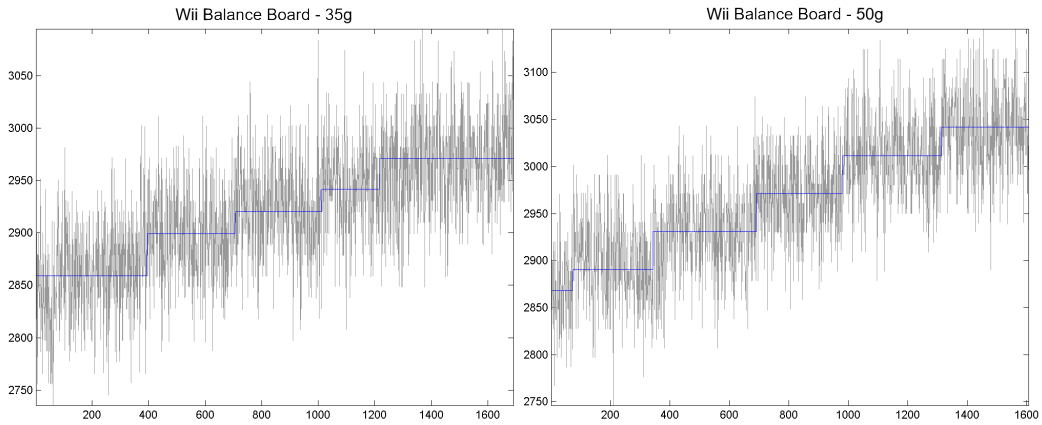


Figure 3.25: Results of sensitivity test for the Wii Balance Board, noting that while extremely noisy, analysis can be performed to reliably determine the points at which 50g weights are applied to the device, but can fail at a weight of 35g. *X* axis is time, *Y* axis is reported weight in *g*.

Weight	Mean	Error	% Error
20g	60.996	40.996	204.982
100g	94.695	-5.305	-5.305
500g	288.638	-211.362	-42.272
1000g	604.043	-395.957	-39.596
2500g	1971.451	-528.549	-21.142
5000g	4473.207	-526.793	-10.536

Table 3.10: Wii Balance Board force calculation results. *Mean* column is mean value across all recorded samples at each weight, measured in grams.

Weight	Mean	Error	% Error
20g	21.74	1.74	8.68
100g	113.59	13.59	13.59
500g	587.22	87.22	17.44
1000g	1061.38	61.38	6.14
2500g	2489.76	-10.24	-0.41
5000g	4743.65	-256.35	-5.13

Table 3.11: Saitek X-65F Force Calculation Results at 10.0kgf. *Mean* column is mean value across all recorded samples at each weight, measured in grams.

3.7.3.3 Saitek X-65F

The Saitek control software provided with the joystick allows for the setting of a number of variables, including deadzone per axis, and maximum reported force. This latter ability necessitated a number of additional tests to see how the maximum value related to the accuracy of output.

Before running the experiments with the Saitek device, a sample of the device at rest was recorded, with the maximum reported force set to 10.0 kgf. This revealed a mean x-axis position of -22.54 (stdev 1.24), and y-axis position of -9.48 (stdev 1.45), in the native units of the device. The -22.54 offset on the axis equated to the mechanism being off by approximately 0.1 kgf. By leaving the joystick alone and changing its force setting to have a maximum of 0.2kgf movement the offset rose to approximately 50% inaccuracy at the lowest setting. This points towards a physical limitation of the mechanism within the joystick, possibly some inherent bias in the device, as it was also noted that the device would not always 'centre' correctly when pushed, similar in effect to the accelerometer issues discussed in Section 3.2. Table 3.11 shows the results of the accuracy experiment when the device software is set up to have a maximum reporting value of 10.0 kgf in the x and y axis. As can be seen, the values reported are accurate only to within 10% or worse within the range of values the devices are expected to handle when used to detect human force. As the device supports a variable maximum detection value, a further accuracy experiment was performed with a lower maximum value of 2.5 kgf; the results of this are collated in Table 3.12. It had been hoped that at a lower range, the

Weight	Mean	Error	% Error
20g	22.97	2.97	14.84
100g	117.91	17.91	17.91
500g	580.1	80.1	16.02
1000g	1131.43	131.43	13.14

Table 3.12: Saitek X-65F Force Calculation Results at 2.5kgf. *Mean* column is mean value across all recorded samples at each weight, measured in grams.

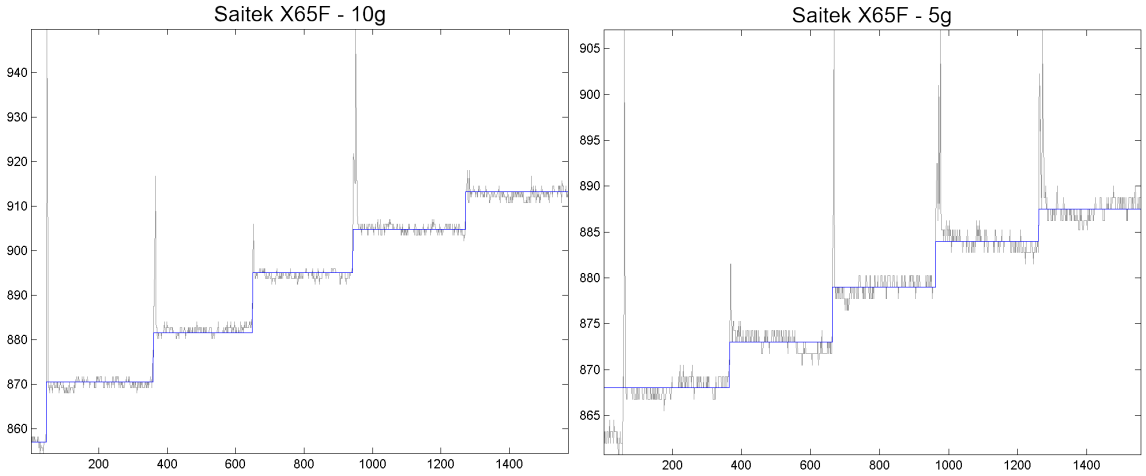


Figure 3.26: Results of sensitivity test for the Saitek X65F. The device was capable of reliably detecting 10g and 5g weight increments, with low noise.*X* axis is time, *Y* axis is reported weight in *g*.

reported values would be more accurate, but this was not the case. In the sensitivity experiment, the joystick was tested with a maximum setting of 2.5kgf in the Saitek control panel, and was shown to perform well, being able to detect each successive weight correctly at 10g (Figure 3.26 left), and detecting all but the first at 5g (Figure 3.26 right).

It had been assumed prior to the experiment that the device would have good sensitivity, due to its use as a high-end gaming joystick, where accurate and timely response to user input is important, and this was shown to be the case, exhibiting low noise throughout the experiment compared to the other commercial entertainment device being tested, the Wii Balance Board.

The experiments show that the device is capable of reasonable accuracy and sensitivity, making devices based on the same technology as the Saitek X65F suitable for interaction with a range of therapies requiring force detection; the ergonomics of the X65F itself pose a problem, however, requiring gripping a joystick shaft, rather than merely applying force to flat sensor pads, as with the other devices.

3.7.3.4 LoadStar iLoad Mini

The iLoad device displayed a consistently accurate result across all of the tested weights, except for the smallest, which resulted in a -13% error, but even this was a small error in terms of the absolute force applied. Table 3.13 shows that the device was within 3g of the true value at everything but the largest of the weights, suggesting that the device would be capable of accurately recording small changes in force, making it suitable for comparisons of strength and dexterity across multiple therapy sessions. That the device displays little bias or noise is of added benefit, as it means the device can be used over a long play session without reported values drifting, and that little filtering would be required to determine with high accuracy the force being applied in real-time.

Table 3.27 shows the results of the sensitivity experiment. As was to be expected from a device designed for accurate measurement in an industrial application, the sensitivity is excellent, successfully detecting 9 out of 10 5g increments in applied force (left); at 2g the device failed to determine 6 of the 10 weight increments, but this was still significantly lower than the other tested devices.

Weight	Mean	Error	% Error
20g	17.364	-2.636	-13.179
100g	97.657	-2.343	-2.343
500g	499.813	-0.187	-0.037
1000g	997.526	-2.474	-0.247
2500g	2497.161	-2.839	-0.114
5000g	4974.414	-25.586	-0.512

Table 3.13: iLoad Mini Force Calculation Results. *Mean* column is mean value across all recorded samples at each weight, measured in grams.

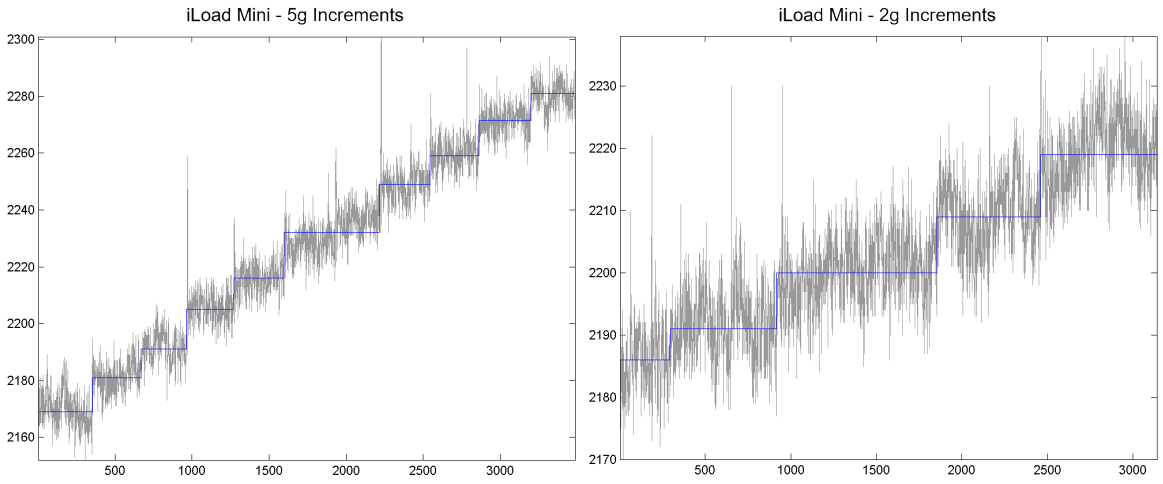


Figure 3.27: Plot of sensitivity results using iLoad device. 5g could be detected reliably, while 2g weights resulted in missed changes. *X* axis is time, *Y* axis is reported weight in *g*.

Weight	Mean	Error	% Error
20g	-39.61	-59.61	-298.03
100g	146.32	46.32	46.32
500g	521.17	21.17	4.23
1000g	798.46	-201.54	-20.15
2500g	1922.2	-577.8	-23.11
5000g	2628.66	-2371.34	-47.43

Table 3.14: PPS C500 Force Calculation Results. *Mean* column is mean value across all recorded samples at each weight, measured in grams.

3.7.3.5 Pressure Profile Systems ConTacts C500 - DLP-IO8

The results of the accuracy experiment using the C500 device are collated in Table 3.14. It was expected that the device would perform similarly to that of the iLoad device, but it was significantly poorer. This could be due to a number of factors. Resistive sensors like the iLoad and C500 require recalibration to prevent a drift in reading; the C500 sensors were older than the iLoad sensors, which were themselves not far from the manufacturer's suggested recalibration date, suggesting that the C500 was likely due for recalibration. Additionally, unlike the iLoad, the C500 is not a complete system; the iLoad comes with a device which converts the iLoad sensor readings into a calibrated weight value, whereas the C500 utilised a third party external device, the DLP-IO8. The DLP-IO8 datasheet [DLP] makes no guarantees as to the accuracy of its analogue readings or of the quality of the 5v line fed to the sensors, so it is certainly possible that the data board was simply not able to produce the best results using the C500 sensors. Such a result should not necessarily be seen as a failure of the board or the sensors, but as an indicator as to the difficulty in providing bespoke solutions for human interactivity.

The noise inherent to signal received from the C500 device is likely the cause of the poor sensitivity test performance, shown in Figure 3.28. At 50g, all of the increments were detected successfully (Right), albeit with an extra notch, which could either be due to noise, or a product of the upward trend of the signal bias. At 35g increments, it was not possible to reliably determine at which point increments occurred (Left), with multiple erroneous steps present.

3.7.4 Discussion: Force Sensing Devices

The ability to detect the movement of a player is only a partial solution to the goals of stroke teletherapy; a complete solution would benefit in some way for the user's strength to be measured. Traditional therapy assessment has made use of expensive standardised hand held dynamometers for determination of user strength [PTA+01] [STBH89], so the ability to replicate this in the home is beneficial. For those suffering from acute stroke, an accurate and sensitive measurement of this force may

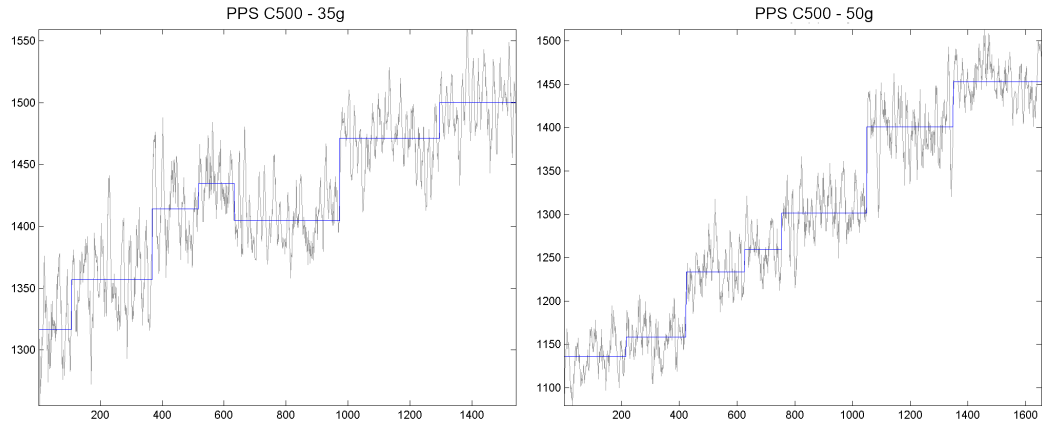


Figure 3.28: Plot of sensitivity results using PPS C500 device. 50g increments can be determined relatively accurately, while 35g begins to show deviations. *X* axis is time (s), *Y* axis is reported weight in *g*.

be the only way in which the affected limbs can interact with rehabilitative software. This chapter has investigated how a range of commercially available force sensing devices perform in three important metrics: their accuracy, their sensitivity, and their propensity for bias.

The results of the accuracy experiment on the chosen devices are collated in Figure 3.29. As was to be expected, the iLoad device, stated to be usable for industrial applications performed better than those for entertainment purposes, generally achieving an accuracy within 1% of the load applied to them. The relatively poor performance of the Wii Balance Board in all of the experiments is an indicator as to what can be expected of commodity force sensing devices, suggesting that as yet there is no off-the-shelf design capable of accurate detection of changes in force. While the Saitek X65F performed to an acceptable standard, it costs over £250, making it much more expensive than any of the motion tracking devices discussed earlier in the chapter.

The results of the bias experiment are collated in Figure 3.24, showing the bias as a percentage of their total stated measurement range. All of the devices display relatively little bias over time, demonstrating their suitability to be used in a rehabilitative context for an extended period of time without requiring a rest or recalibration stage that could be to the detriment of the treatment.

Finally, the sensitivity experiment results are shown in Table 3.15, again showing the minimum detectable change as a percentage of the maximum value the device is able to output. The previous work of Clark et al [CBP+10] had previously described the Wii Balance Board as likely to be poor at this metric, and this has been shown to be correct. The device may however still be useful for rehabilitation of balance and the lower limb. As with the other experiments, the iLoad device excelled, being

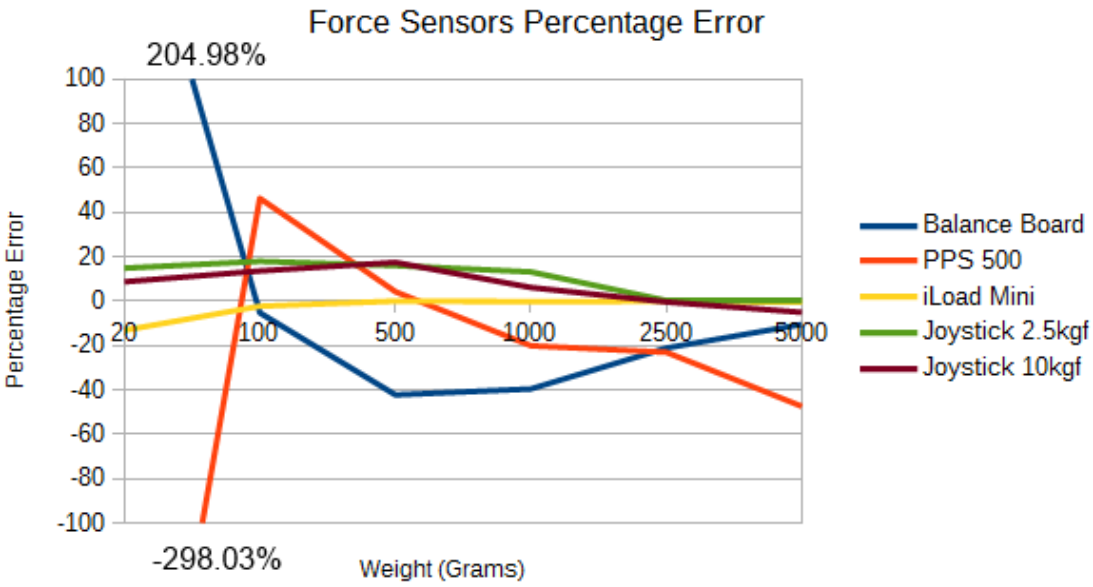


Figure 3.29: Graph displaying error in sensor readings as percentage error from baseline value

able to reliably detect force differences of 5g weights, with some degree of accuracy down to 2g, suggesting that in cases where the accuracy of force applied is important, a device like the iLoad would be ideal.

As part of physical therapy, manual dexterity is often tested via requesting that the participant write something on paper using a pen or pencil, with grading being a subjective assessment of their ability to legibly and smoothly replicate the requested words in a timely manner. There has been work performed on utilising force-sensing pens to detect the uniqueness of someone's signature [DYC+10], suggesting that force sensing devices could be used to create alternate tests of dexterity, that could be assessed objectively. Reynaerts and Brussel [RB95] found that it is common to exert a force of approximately 2 Newtons when writing, with the applied force rarely exceeding 5 Newtons, or approximately 0.51 kilograms of force. For this reason, the determined sensitivities of the devices tested are also stated in terms of a percentage of this value - only the iLoad device would be capable of detecting a force-based task such as writing a signature to within an error of less than 10%. However, for less precise tasks involving picking up or or moving weights of 50g or more, all of the devices would be capable of providing some metric as to patient ability.

3.8 Conclusion

The work in this chapter shows that commercially available devices are capable of producing data output of sufficient quality to to allow both accurate complex mo-

Device	Sensitivity (g)	As Percentage	Percentage of 5N
Wii Balance Board	50	0.0334	98.039
Saitek X65F	10	0.4	19.608
PPS C500	50	0.276	98.039
iLoad Mini	5	0.0276	9.804

Table 3.15: Device sensitivity summary: *Sensitivity* is minimum weight change in grams that could be accurately detected, A*s Percentage* is sensitivity as percentage of device maximum.

tion detection, and an accurate reading of the amount of force a user can impart using their limb. Together, these capabilities can be used to enhance the quality of interaction between a user and a potential piece of rehabilitative software. This is of particular use to packages designed to aid in the rehabilitation of stroke, including acute cases resulting in limited limb movement. The ability to accurately determine the positions positions and orientations of the user's limb can be used to allow software to determine whether a patient is performing their physical rehabilitation moves correctly, and provide feedback to therapists and patients as necessary to ensure that the patient receives the best possible outcome from their therapy.

The position and orientation metrics used to measure the quality of the motion detection devices are universal, and will allow for further comparisons to be made against any new motion tracking devices that are produced, such as the LEAP Motion, or Sixense STEM system, allowing informed decisions to be made on hardware to target when developing motion sensing rehabilitation software.

With the appropriate devices, the patient's limb strength can also be accurately measured, such that even small gains in ability can be reliably measured - increases in limb strength and dexterity over time can result in better scores in 'quality of life' measurements, such as those outlined in Chapter 2. For patients suffering acute stroke, and who may only have limited movement in the upper limb, control of software via detection of force may be the only way in which interaction can be reliably achieved, and so the ability to accurately detect changes in force using the appropriate hardware allows for therapeutic software be designed around this input method.

The work performed in this chapter influenced and informed the work done in the following two chapters, which outline a rehabilitative gaming toolkit suitable for creating software for acute stroke rehabilitation games, and a software package that allows for the creation and testing of gestures in 3D space, allowing therapists to create interactive rehabilitation routines for their patients.

Chapter 4

Detecting Success Markers In Rehabilitative Games For Acute Hemiplegic Stroke

4.1 Introduction

The previous chapter has shown that commercially available motion and force sensing devices have a level of accuracy suitable for use in a rehabilitative software package. However, how best to integrate patient input into the software so as to provide maximum rehabilitative benefit is still to be determined; which motion tasks to ask the player to replicate, how best to determine player performance, and even how best to convey the movement requirements to the player are still open questions within the literature.

As part of an ongoing therapy, a fully integrated rehabilitative game may require the user to attempt to replicate complex actions, both to ascertain a user's current ability, and to facilitate motor learning. A home-deployed system should ideally also be capable of efficient and practical interactions with the user without any trained therapists on-hand to guide the process. It is therefore vital that such rehabilitative software be carefully configured to ensure that the patient can intuitively receive an optimal outcome from the software. As a patient's condition improves, certain movements will become easier, and thus the patient should be asked to perform more difficult tasks, both to further push the rehabilitative benefits, but also to stave off the effects of boredom, which may negatively impact progress [BDRG10]. For rehabilitative gaming software to correctly determine patient progress and provide relevant feedback to the patient, it must be capable of sampling and analysing patient interaction with a high degree of fidelity, to gain the most accurate representation of patient ability possible. The success rate of an in-game rehabilitation routine should be representative of 'real life' gains in motor control; that is, in game

performance should correlate with manually tested performance metrics such as the CAHAI and Wolf tests outlined in Chapter 2. In this way, performance within a video game delivered therapy should be an indicator as to the success or otherwise of a patient's current rehabilitation therapy.

In order to investigate both how best to convey rehabilitative tasks and resulting performance to the user, and how best to analyse user performance, a proof of concept system has been developed, comprising of a customisable video game framework suitable for creating simple rehabilitative test games that can be interacted with via interfacing with force-sensing hardware, and a test game built upon this framework designed to test movements that place importance on accuracy and reaction times. The system was developed with the requirements of those suffering the side effects of acute hemiplegic stroke, with focus on the upper limb. To this end, the game is designed to be played with sensitive force detection devices such as those outlined in Chapter 3, so as to allow the user to play the game even in cases where there is little strength or accurate motor control in the limb.

This system has been used as part of a larger investigative project into the ability of video games to assist in stroke rehabilitation, to demonstrate a link between explicitness of instruction and overall player performance, and to determine whether a video game system is sensitive enough to measure and detect motor learning of sequentially linked actions, as many common day to day activities such as buttoning shirts and opening doors are comprised of many smaller actions that must be performed as one cohesive action - ability at such actions is therefore a likely indicator of a patient's independence, and therefore quality of life. Where necessary, work performed by others relating to my system, including changes and improvements made to the system on their behalf, will be discussed and accredited as necessary. These success indicators, or response markers, could in the future be used to evaluate the efficacy of a routine in aiding patients, and allow for earlier changes in a therapy if required.

The purpose of this chapter is to provide a detailed insight into the system, including its construction, design considerations, and ability to support a range of rehabilitative experiments; which together allowed for rapid prototyping of experiments to quickly obtain high quality results. Results of user trials undertaken using the system are also discussed, including how the system has shown a link between the nature of instruction given to participants and their overall ability to complete them. Preliminary results of trials with post-stroke participants are also shown, which indicate that the software described in this chapter is sensitive enough to detect motor learning even in a paretic limb.

The work in this chapter is an extension of previously published work, with *Bespoke Video Games to Provide Early Response Markers to Identify the Optimal Strategies for Maximizing Rehabilitation* and *Early Response Markers from Video Games for Rehabilitation Strategies* providing a basis of the research goals, and *Adopting Best Practices from the Games Industry in Development of Serious Games for Health* providing insight into the gameplay decisions made to aid in meeting those research goals.

4.2 Rehabilitative Response Markers

The primary goal of the system described in this chapter is the detection of *response markers*. These are detectable changes in physical or neurological condition that indicate that a given medical intervention is having an effect, either positive or negative. Examples of response markers examined within the wider medical literature include degree of olfactory dysfunction as an indicator of response to pharmacotherapies for Alzheimer's disease [VL09], and a specific protein generated as a side effect of intake of drugs for treatment of multiple sclerosis [GMS$^+$08]. Reliable detection of a response marker via analysis of a player's input and task performance within a video game could be used as an indicator of the efficacy of the rehabilitation strategy being employed to aid them, both by their guiding therapists, and potentially within the software itself.

How best to facilitate the detection of response markers, and how to create the response itself are the primary drivers of the modular nature of the presented system, with development of the system occurring concurrently to the test trials utilising it being performed with the Institute of Neuroscience at Newcastle University, investigating both response markers and linked action motor learning.

4.3 Rehabilitation Game Framework

To create a rehabilitation game with which to investigate motor learning mechanisms and rehabilitative response markers, a bespoke, lightweight game framework was developed, upon which such games could be prototyped. The framework provides functionality required for the creation of simple rehabilitation games, such as communication with the force sensing devices, logging of high quality spatiotemporal data, and the means by which to create and visualise a 3D environment, display on-screen information such as the player's score, playback of audio files, as to create an easily modifiable game utilising a stack of game states to control game logic.

The framework, and the game built upon it, were written in the C++ programming language, with Microsoft Visual Studio used as the programming environment. The choice of C++ was primarily due to its compatibility with a wide range of libraries written with C and C++ APIs, including the device driver libraries used by several of the force sensing devices, as well as easy access to the serial port for those devices requiring it for communication. Other benefits include more explicit control over screen refresh rates and the threading model, familiarity with the language, and the ability to reuse previously written code for auxiliary functions such as graphics rendering and audio output.

4.4 Patient Input

For a rehabilitative game to be a truly interactive experience, it must in some way be able to gauge the player's ability to replicate a given task. For the purposes of the research the framework was designed to facilitate, it can be assumed that the player is suffering from the side-effects of hemiplegic stroke, and thus has a weakness in one side of the body. The investigations focus on upper-limb therapy, potentially in those suffering acute symptoms, whereby the user may have extremely limited movement in their effected limb. To ensure that even those suffering acute side effects could take part in research trials, and to investigate the viability of creating rehabilitative software for such people, it was necessary to design the framework's handling of input devices and data logging such that therapy games designed around improvements in dexterity and reaction speed of the hand could be created, with the only limitations on participant ability being movement that can be physically recorded by the chosen input devices.

4.4.1 Calibration

The effects of hemiplegic stroke upon a person's limb are not binary; there can be a range of debilitating effects to motor control. Age, and the general fitness level also play a part in determining the amount of movement and strength in a hemiplegic limb [Sch07][KHJ+14]. To accommodate this, it was necessary to design into the framework a method of calibration to determine the user's ability to impart force upon the input devices, and to utilise percentages of this value within the game's logic, rather than absolute force metrics. To illustrate the importance of this, consider a very simple game: impart 10 Newtons of force upon the force sensor to win. Such a game tells us nothing about a user's current ability, nor how difficult the task is likely to be: an adult may find the game easier than a child, even when taking into consideration the debilitating effects of stroke.

In order to provide a similar gameplay environment across all trial participants, and provide a more comfortable gameplay experience, all user input response is designed around a user's *relative* ability; that is, rather than have a task requiring the player to enact a force of 10 newtons on a sensor, they must instead push with a force proportional to a previously recorded 'maximum' calibrated force. By using this concept of relative ability, a game built around the framework's capability is as suitable for a small child as it is for an adult. In order to determine this maximum capable force, the framework supports a calibration stage to present to the player at the start of the game, whereby the player is required to push down on each sensor a number of times as hard as they comfortably can. The mean result for each sensor is then used as a baseline measurement for future in-game tasks, where a player may be asked to push the sensors to some percentage of their previously calibrated ability.

4.4.2 Hemiplegic Considerations

The debilitating effects of a hemiplegic stroke are likely to result in a difference in ability between the user's affected and non-affected limbs, presenting as a change in dexterity and overall limb strength. It is necessary to take this into consideration within the game framework when dealing with user input. As part of this, the initial calibration of sensors is performed using both of the user's limbs, and calibration values for each maintained separately. This intuitively allows the game to maintain a relative 'difficulty' between games using the able and paretic limbs - a task may be programmed to require 20% of a user's maximum force to be applied in order to pass, which may be 5 or 20 newtons, depending on the limb to be used to complete the action. The framework additionally supports run-time selection of the current hand to receive input from as part of the configurable aspects outlined more fully in Section 4.7, allowing parts of games built upon it to target specific limbs - this could be used to present gameplay tasks less often to the paretic limb, so as to reduce the affects of fatigue.

4.4.3 Tare Weight and Sensor Drift

When dealing with force sensors held or otherwise activated by the hand, it is likely that some amount of force will be enacted upon the sensor at all time, whether from the grip of the user, or as a side effect of their hand resting on the sensor. Therefore it is desirable to take this into consideration when calculating how much meaningful force the user is actually imparting on the sensor. During a play session, there are designated 'rest periods' where the user is requested to simply place their hand on the sensor (or grip it gently) so that a recording of the mean force reported by the sensor can be taken. This serves two purposes; firstly it removes the *tare weight* of

the user's hand or grip from the sensor value, and secondly it reduces the effect of *sensor drift* - as ambient conditions such as temperature and humidity change over time, it is common for the baseline value reported from a sensor to drift somewhat, requiring correction to maintain the accuracy of the force values recorded by the framework.

4.4.4 Normalisation of Input

As well as the need to work with a wide range of user ability and to deal with shifting value ranges due to recalibration, the framework must provide a way of handling a number of different input devices, each of which may report force in their own format. This can be neatly handled by having all aspects of the framework dealing with a user's input deal with a *normalised* value, where 1.0 is the maximum calibrated value, and 0 is considered 'no desired input'. The normalised value at any given moment is calculated as follows:

$$Force = \frac{Input - T}{U} \tag{4.1}$$

Where U is the user's calibrated maximum force, T is the latest tare reading, and $Input$ is the latest sensor reading; all variables are in the device's native unit of measurement.

This normalisation simplifies the process of integrating new force sensing devices to the framework, as games built using it do not need to be programmed with *a priori* knowledge of the connected force sensor's capability or reported value range, only that the input will always range between 0.0 and 1.0 inclusive. This equation also neatly accommodates the wide range of user abilities, by allowing all further inputs to be described in terms of a normalised ratio.

4.5 Hardware Device Usage

The framework as a whole is designed to theoretically support any number of different force sensing devices; all code within the system targets an abstract base class that specific force sensing device implementations can be derived from. Throughout the research conducted with the framework, three devices of different capability, accuracy, and price, were tested and utilised in research trials. Their hardware capabilities are discussed in detail in Chapter 3; note that the Wii Balance Board was not considered suitable for the trials, due to its poor performance at detecting slight changes in force. Due to differences in their physical setup, there a number of considerations to take into account when utilising each of the devices for trials.

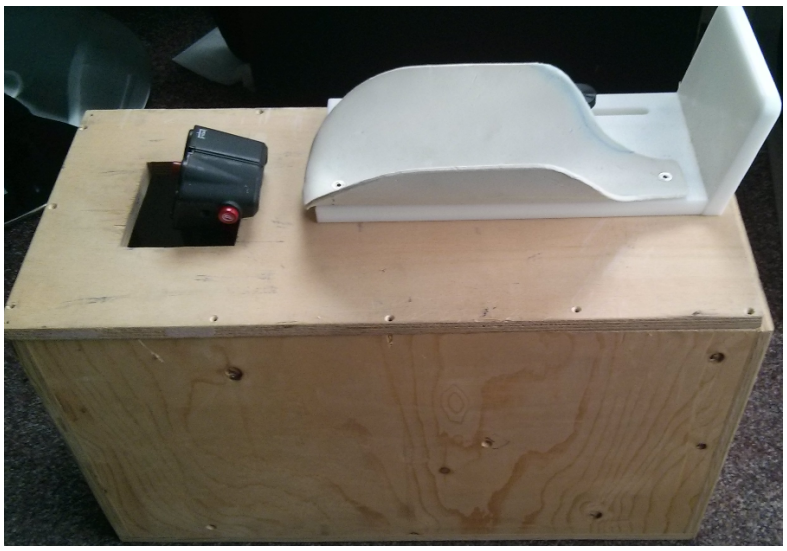


Figure 4.1: The Saitek X-65F device within the wooden enclosure used to allow the users forearm to be in-line with the top surface of the joystick.

4.5.1 Saitek X-65F Combat Control System

The first device integrated into the framework was the Saitek X-65F gaming joystick; this was chosen due to its ease of availability as a commercial gaming peripheral, the ability to report user applied forces in PSI and KGf, and relative affordability in comparison to dedicated force sensors. As the users of the system could suffer from a range of disabilities that could influence the range of motion of their upper limb, the Saitek joystick was deemed unsuitable for use in its standard setup. Rather than being placed upon a desk with the hand gripping the shaft, the joystick was instead placed inside an enclosed wooden box, with the top of the joystick protruding through a hole in the top; additionally the joystick was rotated around the shaft by 180° so that a flat surface faced the user rather than the buttons and POV switches. Together, this resulted in a much smaller visible device, suitable for resting the hand on to push as desired 4.1. Two such bespoke configurations were constructed, allowing a modified joystick setup to be mounted on either side of a chair, allowing both hands to be used interchangeably during gameplay.

4.5.2 Pressure Profile Systems ConTacts C500 and LoadStar iLoad Mini

In additionto the Saitek joystick, a pair of more traditional force sensing devices were also tested. These both took the form of small sensors attached to a control unit; the PPS C500 devices consist of small fabric pads, while the LoadStar iLoad Mini devices are small metal cylinders. These devices were chosen due to their small form factor, allowing for direct placement in relation to the user's hand for optimal force application. The C500 was used initially, but concerns for cleanliness in the long term led to using the all sealed, all metal iLoad device, as it could be eas-

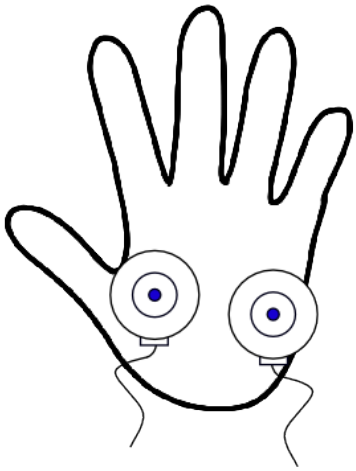


Figure 4.2: Placement of PPS C500 and LoadStar iLoad Mini sensors under the user's hand when playing games utilising the Framework.

ily sterilised. Both devices are tested for their ability at accurately detecting force in Chapter 3.

Both sets of force sensors utilised a similar setup to each other. At the start of a user's game session, sensors were taped to the user's palms using medical tape, at positions aligning to the *digiti minimi* and *pollicis brevis* muscle groups, as seen in Figure 4.2. By tensing these muscles (used to bring the thumb and little finger inwards, respectively) or rocking the hand, it was possible to impart force upon one or both devices attached to each hand. This allowed effective simulation of an axis of movement similar to a joystick - pressing on the right sensor increases the magnitude of positive movement, while the left increases the negative movement magnitude. The PPS C500 system was easy to set up in this manner, being made of fabric squares with no 'sidedness'. The LoadStar iLoad mini devices were actually attached upside down on the users palm, to maximise adhesion surface area, meaning that the 'stud' of the device was touching the tabletop surface.

4.6 Overview of Framework

A game can be seen as a virtual environment in which player interactivity takes place, viewed from a specific point in the environment - through the character's eyes in a first person shooter like *Quake* (www.idsoftware.com; *Zenimax Media Inc.*), or from top down in a game like *Asteroids* (*Atari, Inc.*). The framework detailed in this chapter works as a small but functional game engine capable of supporting the development of such games, and is designed with knowledge of engine fundamentals in mind. It has been designed in a generic fashion in which an array of simple games could be created, and has been designed with the concept of a clear separation between 'engine' and 'game' to allow as much code reusability as possible -

the engine provides features such as rendering, sound output, and input handling, to a 'game' class that creates and controls these features.

The framework allows the creation and control of 3D scenes in which gameplay takes place, as well as graphical overlays to inform the user. Elements within these 3D scenes are manipulated by game-specific logic, possibly in reaction to user input. Audio can also be played as necessary. These scenes are viewed through a controllable 'camera', that can be positioned and orientated in 3D space, with a specified field of vision. The framework is comprised of a number of classes encapsulating the functionality required by the games. Many of these classes are designed to have concrete implementations derived from them for each game. A high-level overview of the classes making up the framework is provided in Figure 4.3. The core of the framework is a number of 'systems', handling windowing, rendering, sound output, and gameplay. Each system is created upon startup, and is accessed using a *Singleton* design pattern.

4.6.1 Asset Management

Data files stored on disk used to correctly represent the 3D environment such as textures, sound files, and mesh geometry are considered 'assets'. Each asset type is handled in the framework via a 'Manager' class for each type, which regulates file access and memory footprint to ensure that files are not loaded repeatedly. For example, the following code is used to load some geometry from a file for rendering as part of the 3D environment of the game:

```
Mesh* t = MeshManager::Instance().LoadMesh("asteroid.obj");
```

If the file has been previously loaded, a pointer is returned to the loaded data class, otherwise the file is loaded, processed, and the appropriate class returned. Managers are responsible for ensuring the consistency of the program at runtime; therefore if a file fails to load, a Manager will return a 'default' representation of an asset; in the case of geometry this is a simple cube, as it is trivial to generate without requiring data from a file, and will allow program execution to continue without explicit null pointer handling.

4.6.2 Graphical Rendering

The framework provides the functionality required to draw both 2D and 3D graphical elements to the screen. All graphical drawing (also known as *rendering*), whether 3D 'world' geometry, text, or menus and logos, is produced using the *OpenGL* graphical API, a cross-platform library that is almost universally supported

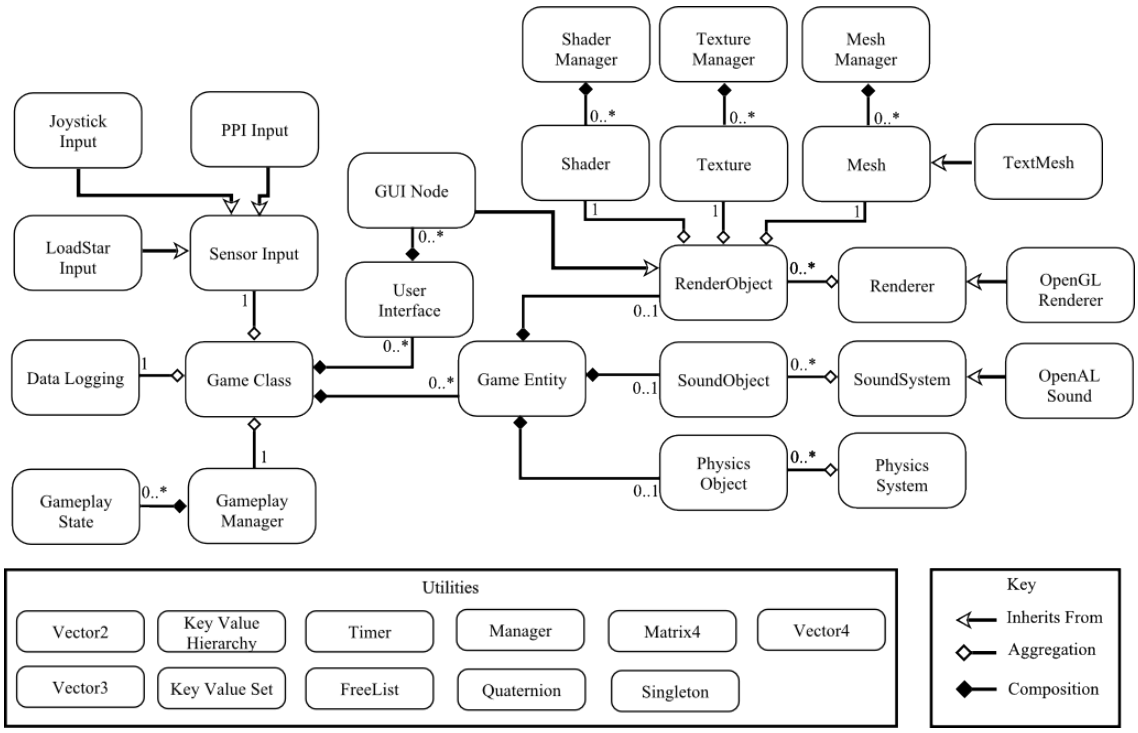


Figure 4.3: Class Overview of Framework, showing relationship of major components

by consumer level desktop and mobile graphics hardware. Modern graphical rendering generally requires three pieces of data for each drawing operation: some geometry to draw (often known as a *mesh*, and is made up on many *vertices* - points in 3D space that together form a number of polygons, generally triangles), a set of 1 or more pictures to apply across the surface of the polygons, known as *textures*, and a small program executed by the graphics hardware to process the geometry and texture information into the final desired shape and colour on screen, known as a *shader*.

Within the framework, any images required for rendering are loaded into memory from files using the *Simple OpenGL Image Library* (www.lonesock.net/soil.html; *Johnathan Dummer, Sean Barrett*), a popular C library capable of loading and transforming common image formats such as *PNG* and *JPEG* into RGBA data suitable for use within OpenGL. Shader programs are generally stored in plain text files, with the host system's OpenGL driver being responsible for their real time compilation into assembly suitable for executing on the graphics hardware of the system. Mesh data is converted from file to OpenGL vertex buffer data by custom code, and is assumed to be in one of the following popular mesh formats: *OBJ*, *LWO*, or *MD5*. The rendering system is also responsible for the drawing of text on screen, with fonts transformed from their native vector-based *truetype* format to standard texture data using the *Freetype* C library (www.freetype.org; *The Freetype Project*).

4.6.3 Audio Playback

In-game audio playback is produced using the *OpenAL* (www.openal.org; *Creative Technology Ltd.*) audio hardware abstraction library, a well supported cross-platform library controlled by Creative Labs. The library requires raw pulse code modulated data in memory, which is then processed and played either directly to the speakers, or processed to simulate a 3D environment, with panning and volume attenuation based on distance. Audio file assets must either be in the *WAV* or *OGG* formats, with WAV files being loaded into memory in their entirety, and OGG files being streamed in from disk as necessary, making them an efficient solution for music playback.

4.6.4 Game Creation

In order to create an actual playable game using the framework, a specific new class must be made, which derives from the *GameClass* base class provided by the framework. This class should then control a number of *Entities* - classes which represent an intractable object within the game world. These entities have a transformation matrix encoding its position, rotation, and size within the game world, and can have optional graphical and audio elements. Entities are arranged in a tree structure - child entities have positions, scales, and orientations relative to their parent entity, allowing complex scenes to be built and moved as necessary. Each entity class may have its own derived *Update* function, which will be called upon every frame of the game, allowing game logic to be created via the interaction of specifically crafted Entity elements.

The game itself consists of an endless code loop, with each iteration forming a single game 'frame', within which game entities are updated, game logic executed, audio cues played, and graphics rendered to the screen. The time taken to execute and render a frame is used to correctly integrate game logic such as movement and animation, so that the game behaves the same across all machines, regardless of how quickly frames are processed.

4.6.5 Game Execution Using State Machines

The runtime execution of games written on top of the framework is managed by the use of a number of *State* classes, which are stored and accessed as a *stack* data structure. Only the *State* at the top of the stack is updated every frame, and during its update may choose to either **push** a new *State* onto the top of the stack, or **pop** itself off, allowing the previous *State* to update from the next frame onwards. This mechanism, often known as a *pushdown automata*, allows for intuitive handling of logically separate functionality: A 'main menu' state may push an 'options menu'

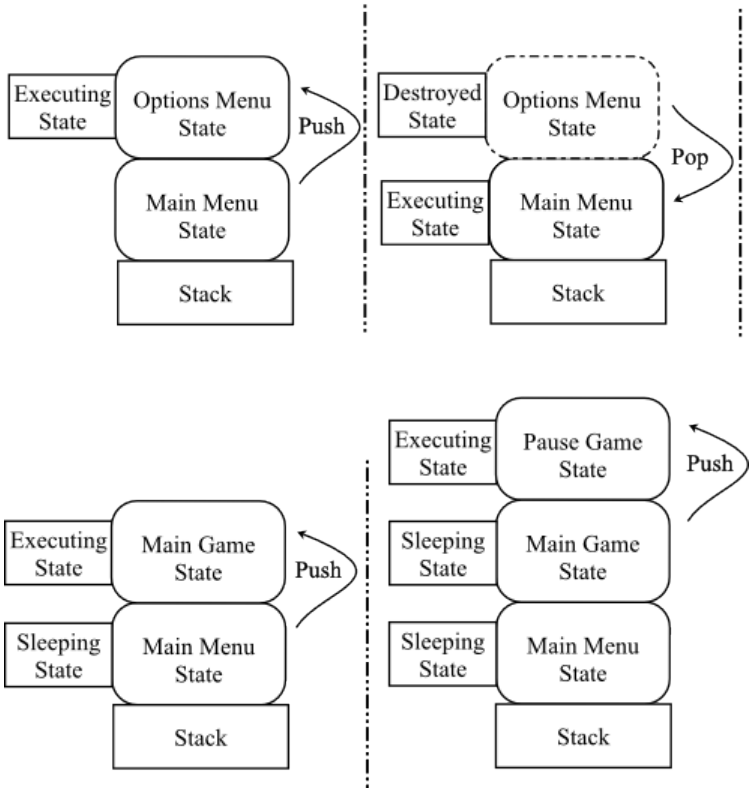


Figure 4.4: Stack Based Game Updating: Only the top state updates every game tick; *pushing* a new state will set the previous state into a sleep state to be woken after a later state *pop*.

or 'main game' state, which may further push a 'pause game' state when a button is pressed, which can then be popped off the stack when gameplay is to resume. Encapsulation of code in this manner promotes code reuse, allowing multiple games to be developed from core 'building blocks' of code.

State class instances are informed of changes to their stack that influence them, such as being promoted to the top of the stack, in order to allow cleaning up of temporary resources via virtual functions called when such an event occurs; a newly promoted *State* class is said to have been 'Woken up', while an active *State* having a new *State* pushed above them is said to have 'Gone to sleep'. A *State* encapsulating 'main menu' functionality entering a sleep state due to a new *State* being pushed may for example remove the graphical elements that make up the menu, and automatically re-add them later if the new state is popped off, triggering its woken state.

4.6.6 Input Handling

The force sensing hardware utilised by the framework is represented within the code as a number of individual 'devices', each with up to two axis of movement, with the current axis state stored as a floating point value. This affords easier compatibility

```
TopLevel:Root
{
        InnerLevel:Child
        {
                KeyExample:True
                Duplicate:Text Entry
                Duplicate:0,0,0
        }
}
```

Figure 4.5: Data file hierarchy and contents: The inner node is considered a child of the *Root* node, while the *KeyExample* key is considered the data of the *Child* node

between device types within code; even though the iLoad Mini is physically 4 sensors, it is easier to treat them as two pairs of axis, similar to a pair of joysticks.

To ensure that the current state of the connected devices is calculated as quickly as possible, interaction with the any connected hardware happens in a separate thread to the main game execution. Due to the importance of accurate and timely sensor readings, this thread never sleeps or yields - if necessary it will enter spinlock until new data has been read from the device. To communicate with the main game thread, data is updated atomically, by copying of cache-aligned structs from thread-local storage to the main input class, allowing for error-free updating without requiring explicit locks.

4.7 Configurability

One of the main benefits of the presented system is the ability to rapidly reconfigure and re-execute the software. This is achieved by utilising configuration files that control many of the aspects of the system, including device parameters and ordering of the state stack. The contents of these files takes the form of a human-readable sets of attribute-value pairs, arranged in a hierarchical fashion. These files can easily be modified inside text editors, or with a provided bespoke application developed in C++, and utilising the Qt library to provide an easy to use GUI. This format allows those without programming knowledge or the required development software make changes to the system, and the games running on it. This ease of reconfigurability is a step beyond rehabilitative gaming experiments using commercial video games documented in related literature, and has aided in the rapid-prototyping of gameplay specifics when searching for early response markers, as on-site changes can be made prior to user trials, when necessary.

4.7.1 File Format

The data files take the form of a hierarchical data notation, whereby nested pairs of braces denote *nodes*, each of which may have a textual *name* and *type*, and a number of textual key-value pairs. A small example of this format is shown in Figure 4.5. In the example, it can be seen that the 'top level' node has the 'type' *TopLevel*, and the 'name' *Root*, and contains one child, with the type *InnerLevel* and name *Child*. The node *Child* contains three key-value pairs, one of type *KeyExample*, and two of type *Duplicate*; multiple keys are permissible where necessary, with the API providing the means by which to iterate over such multiples.

The format itself bears similarities to the popular data *Javascript Object Notation* interchange format (www.json.org), but with differences in which datatypes can be written or read from the file. Unlike JSON, the parser for the data file guarantees support for comments, using the standard 'C' style commenting format of double backslashes; this allows for easier prototyping of different parameters, and descriptions of such parameters with the files. Any datatype with overloaded stream operators within the C++ codebase may be written to or read from a data file; which when combined with the hierarchical nature of the data notation, allows for a natural representation of a wide range of game constructs within the file.

The class that parses and represents data files internally to the system allows for sub nodes to be iterated over, and node keys to be searched or iterated through. Child nodes of the same type are considered 'siblings', with API functionality for iterating over only siblings of a given type, as well as more general iterators over all siblings of a given node. Nodes and keys may also be accessed directly within the code using a dotted string format, as follows:

```
KeyValueSet set; bool keyResult;
bool found = set.TryKey("Root.Child.KeyExample", keyResult);
```

This string would directly access the key *KeyExample*, from the node *Child*, subnode of *Root*, and place the result inside the variable *keyResult* if possible. This allows data files to be used by the game code in a manner similar to the Windows registry, allowing for a great deal of expressiveness in data representation within data files.

4.7.2 Configurable Aspects

The data files are used throughout the framework to provide configurable data in a common format, reducing program complexity and easing the system learning curve. The main areas of the framework that require configuration changes are those involving the input system, the user interface, and the overall set of states that a game built on the framework goes through during the course of program execution, including any interactable objects that make up the gameplay experience.

4.7.2.1 Windowing and Input

Upon program startup, the data hierarchy contained within the file 'Window.cfg' is always read; this file is expected to be at the same folder hierarchy level as the program executable itself. The first node of type *WindowConfig* is found, and if present the following keys read:

- **Width**: Width of game window
- **Height**: Height of game window
- **Fullscreen**: Is the game window full screen
- **InputType**: String name of desired input type
- **Gameflow**: Filename of game experiment to execute

The *InputType* key consists of a text string, matched against a map within the framework of supported input devices to be initialised, while the *Gameflow* key is used as a file name for loading a further data file describing game specific data structures, outlined in more detail in section 4.7.2.3. By supporting game loading from selectable files, it becomes easier to rapidly develop alternative prototypes of game behaviour without risk of losing useful configurations.

4.7.2.2 User Interface

The hierarchical node system provides a simple method by which graphical user interfaces can be built up and modified, and accessed by game states for rendering. Graphical user interfaces rendered by the system follow a hierarchical *scene graph* structure, common in computer graphics. Each node in the graph represents a graphical element - for a user interface this could be a textured quad to represent a button, or a line of text. Each node has a number of properties relating to how it should be rendered, including a texture, mesh, and a transformation matrix; a 4×4 homogeneous matrix that can represent any standard affine transformation. Any child nodes of an individual scene node are considered to be *attached* to that node, thus performing rendering transformations such as movement, rotation and scaling

```
GUINode:WaitMessage
{
    Position:0,0,0
    Size:720,480
    Texture:Joystick Game/BlueNebula.jpg
    TextNode:LoginText
    {
        Position:10,0,0
        Size:720,300
        Font:5thagent.ttf
        Text:Info
    }
    TextNode:MessageBox
    {
        Position:50,200,0
        Size:620,300
        Font:5thagent.ttf
        FontSize:40
        Text:Hello!
    }
    TextNode:StateTimer
    {
        Position:10,430,0
        Size:200,40
        Font:5thagent.ttf
        Text:Scores
    }
    GUINode:ContinueButton
    {
        Position:425,400,0
        Size:285,70
        Texture:BigButton.png
        HoverTexture:BigButtonHigh.png
        Action:PopGameState

        TextNode:ContinueText
        {
            Position:0,10,0
            Size:285,70
            FontSize:40
            Text:OK!
        }
    }
}
```

Figure 4.6: Data file for a user interface (Left) and resulting graphical representation (Right).

on a parent node affects all of its child nodes also, achieved by multiplying child transformation matrices by that of their parent upon rendering. This is useful for creating simple animated menus - for example a single 'options' node could have multiple child 'button' nodes, which can all be 'slid' onto the screen when pressing a button by simply modifying the screen position of the common parent node.

This hierarchical model naturally maps onto the data file format, allowing intuitive and rapid design of user interfaces, both in order to provide in-game instructions, and to operate as the front end menu system. An example GUI data file, and the resulting on-screen GUI are presented in Figure 4.6.

From a computer graphics perspective, each user interface is rendered by performing a depth-first traversal of the scene graph, and rendering either a textured quad, or a series of textured quads for a line of text, using an orthographic projection matrix to project nodes into clip space. Nodes are always drawn with their position relative to that of their parent. The combination of depth-first traversal and a disabled depth buffer result in children being rendered 'on top' of their parent. Regardless of actual screen resolution, the orthographic matrix used for node projection divides the screen into an area of 1920×1080; this allows for user interface nodes to be placed correctly relative to each other without foreknowledge of the actual screen resolution of the device it will be displayed on.

The user interface system responds to three data node types when reading a valid hierarchical data file:

- **GUINode**:Node representing a graphical element
- **TextNode**:Node representing a line of text
- **OnSignal**:Node representing a set of actions

Of the three, *GUINodes* and *TextNodes* represent things visible on screen (or translations for visible child nodes), and *OnSignals* represent a set of actions that will be enacted upon a parent node if that user interface element is signalled by the framework with a specific string.

A *GUINode* will respond to a series of keys contained within the data node representing them, all of which in some manner inform the rendering system how to produce the correct on-screen visuals:

- **Position**: Position on virtual canvas (float x,y)
- **Size**: Size on virtual canvas (float x,y)
- **Texture**: Filename of texture asset to display
- **HoverTexture**: Texture asset to display with mouse over node
- **HoverSound**: Filename of sound asset to play when mouse enters node
- **Colour**: Colour to tint node visuals (float r,g,b,a)
- **NoMesh**: Do not draw this node (allows for 'hidden' parent transforms)
- **Active**: Whether a node and its children should be drawn or not
- **Action**: Name of C++ function to enact on object when node is clicked on

Additionally, *TextNodes* respond to further keys, which determine how text is to be displayed:

- **Justify**: Left/Right/Center justification of text (string)
- **Font**: Filename of truetype font to display text in (string)
- **FontSize**: Font size to display text in (float)
- **Locale**: Integer index into list of strings (integer)
- **Text**: String of text to display (string)

The *Locale* tag can be used to display a different text string, depending on a selected language. In such cases, a list of strings in a given language are loaded from a text file, with each carriage return denoting the start of another string of text. The Locale tag is then assumed by the framework to be an integer to use as a zero-based offset into this array of strings. It is possible to create a button with both a graphical and textual element, by having a parent GUINode with the appropriate graphic keys set, and an attached child TextNode of the same dimensions with centre justification of text; the render ordering from the traversal noted in Section 4.7.2.2 guaranteeing that the TextNode will be overlaid on top.

The core framework provides a number of C++ functions, mapped via strings, that the GUI system can call when clicked on via the node's *Action* key, such as functions to quit, pause, or restart the game. Further functions can be made callable by the GUI system from the game code ran on top of the framework - functions receive the GUINode that called them as a parameter, allowing extraction of key information as necessary.

The user interface system was designed to balance ease of use against the need for the user interfaces to allow data input and game manipulation. Part of the rationale behind the coordinate system used (with (0,0) representing the top left of the screen, and (1920×1080) representing the bottom right of the screen, rather than the normalised system between -1 and 1 on each axis naturally found within the graphics pipeline) was to allow easy mockups of user interfaces within paint programs such as *Adobe Photoshop* (www.adobe.com/uk/products/photoshop.html; *Adobe Systems Software*) and the *Paint* application bundled with Microsoft Windows (www.microsoft.com; *Microsoft Inc.*), both of which utilise the same coordinate system. This allowed changes to the user interface to be graphically prototyped beforehand, before undergoing the process of manually typing in coordinates. By utilising the same text-based data format both for generation of user interfaces and in specifying the parameters of the game itself (described in Section 4.7.2.3), it was found that less user training was necessary.

The ability to input text directly using the 'Text' tag, rather than via a separate Locale file was the result of user feedback, who found that matching numbers up to strings in a separate file hard to work with, with the limitations placed on the eventual trial participants meaning that support for differing languages not necessary. The ability to sent the font size and typeface was a requested feature, as it was a concern of the Institute of Neuroscience staff running trials using the system that people with impaired vision may have trouble reading the on-screen feedback and instructions, requiring quick and easy adjustment.

4.7.2.3 Gameflow

As described in Section 4.6.5, a rehabilitative game is created atop the framework by creating a number of game state classes in C++, containing the code unique to that game's operation. How these classes are progressed through by the system is determined via loading and parsing another data file, known as the *gameflow* file, loaded at the start of program execution from a file path determined as described in Section 4.7.2.1.

A gameflow file is assumed to be a text file containing data in the format outlined in Section 4.7.1. The file should contain a number of nodes of type *GameFlow*, each of which is further comprised from a number of *State* nodes; together, these form a grouped series of actions to control the logic of the game. Each instance of a *State* node will instantiate an equivalent *GameState* class within the codebase, with the class selected based on the node's name parameter. Whichever *Gameflow* node comes first in the file is considered the 'default' flow which will be started upon program execution, with the first state from the *GameFlow* pushed onto the state stack. During the course of a state's execution, it may push any number of additional states onto the stack, but if the stack becomes empty, the next state from the active gameflow will automatically be pushed, allowing execution to continue.

An example of a *GameFlow* is shown in Figure 4.7. In the example, a basic front-end to a game can be seen. There are three states, matching up to C++ *GameState* classes classes provided by the system - *LogInScreen*, *ZeroSensors*, and *SwitchFlow*. In this instance, as *LogInScreen* is the first named state in the GameFlow, the *LogInScreen* GameState class will be pushed onto the state stack, and its *Update* function will be executed every frame until it deems itself complete, at which point the next GameState class (*ZeroSensors* in the example) will be pushed onto the stack and begin executing. The state *SwitchFlow* is provided by the framework, and will clear the game state stack, and restart execution with the named *GameFlow* node; it is permissible to create cycles whereby a *SwitchFlow* state executes its own parent *GameFlow*. An executing GameState class is aware of both the state node that instantiated it, and the GameFlow node it is part of; allowing for extraction of data from the data hierarchy - each of these nodes may contain as many nodes and keys as necessary to control the logic of that state. In this example case, the *ZeroSensors* node contains two keys, *Message* and *WaitTime*; keys within a State node are assumed to in some way control the workings of the class being enacted on, in this case providing a message and a countdown timer to be displayed using a GUI.

The GameFlow system allows for easy modification of the running of a game. For example it would be trivial to make modifications to a game to test if a player heeds

```
GameFlow:IntroScreen
{
	State:LogInScreen
	{
	}

	State:ZeroSensors
	{
		Message:$PATIENTNAME$, place your hands on the sensors, and wait
		WaitTime:10000
	}

	State:SwitchFlow
	{
		GameFlow:AsteroidFlow
	}
}
```

Figure 4.7: An example GameFlow, and its associated states: The child nodes of *IntroScreen* will be advanced through in sequence, resulting in a new GameFlow being loaded via the *SwitchFlow* state

instructions more carefully when addressed by name by changing a key parameter, or change how many levels in a game must be played by adding more or less of the appropriate *State* entries. The difficulty of a particular movement, or the length of time allowed to complete a movement could also be modified easily using keys within the appropriate state nodes. Such small changes would be possible without support from a programmer, as they would require no additional compilation of C++ code.

4.7.3 Game Entities

When playing a video game, whether it be for rehabilitation or simply for pleasure, the player will be in some way interacting with one or more *entities*; that is, objects within the game that in some way afford an interactive experience, whether it be the player, and enemy, a powerup, or a wall. As with other elements of the game system, entities are loaded into a game via data files, which are loaded and parsed from within the *GameState* classes that comprise a game. Such a data file will consist of one or more *EntityDef* nodes, each of which defines a single type of entity object seen within a game. Figure 4.8 shows an example of such an entity, a space ship that the player controls. An entity within the system code is represented by an instance of the *GameEntity* class, or a class derived from it; an *EntityDef* node may determine which class represents it using the *BaseClass* key - all such *GameEntity* classes within the framework register themselves with a map of pairings to text strings and class constructors upon system startup. An entity usually has some sort of graphical representation on screen, generally taking the form of some sort of mesh geometry, with a texture applied to it using a specific shader program. This graphical representation is determined by a child node of the *EntityDef* node, *RenderNode*. Upon parsing, the following keys are responded to:

- **SceneNodeType**: Class of render representation

```
EntityDef:SpaceShip
{
        BaseClass:GameEntity

        RenderNode:Default
        {
                SceneNodeType:SceneNode
                BoundingRadius:100
                ModelScale:1,1,1
                Material:GameViper
                Mesh:viper.obj
        }
}
```

Figure 4.8: An example Entity Definition: This set of data nodes creates the player spaceship, deriving it from the *GameEntity* class, with the *viper* mesh file used to represent it on screen

- **ModelScale**: Size of mesh on screen. (float x,y)
- **Material**: Texture and shader used in rendering
- **Mesh**: Filename of model to load.

Similar to the *BaseClass* key, the *SceneNodeType* consists of a string, which is paired within the system to a particular class type, which controls the actual rendering of an entity on screen; the system provides *SceneNodes*, for standard 3D geometry, and *ParticleNodes*, for particle effects such as smoke and water. The *Material* entry can either be a child node of the *Material* type, or a standard key parsed as a name of a previously loaded *Material*. Material nodes have the following entries:

- **Shader**: Name of shader to use in rendering
- **TextureLayer**: Name of texture to apply to geometry
- **MaterialValue**: Name and value of a shader uniform

This allows a large degree of control over the graphical representation of an entity on screen, as even assets requiring data to be sent to the graphics card can be defined using the exact same mechanism as all other data within the system. The entity mechanism allows for games employing the system to create their own bespoke entity types for inclusion within the virtual environment, and for easy modification of such entities without requiring development tools.

4.8 Data Logging

In order to analyse the performance of those playing a game, and make informed decisions over likely triggers of response markers, it is important for a rehabilitative

TIME	BLOCK	TRIAL	TargetX	TargetY	CursorX	CursorY	TargetDist
22638.2	1	0	0	0	0	0	0
22770.6	1	0	0	0	-0.00483125	-0.00637895	1.41421
22771.2	1	0	0	0	-0.00483125	-0.00637895	0.008002
22771.8	1	0	0	0	-0.0042938	-0.00713438	0.008002
22772.4	1	0	0	0	-0.0042938	-0.00713438	0.00832683
22773.1	1	0	0	0	-0.0042938	-0.00713438	0.00832683
22773.6	1	0	0	0	-0.0042938	-0.00713438	0.00832683
22774.2	1	0	0	0	-0.0042938	-0.00713438	0.00832683
22774.7	1	0	0	0	-0.0042938	-0.00713438	0.00832683
22775.2	1	0	0	0	-0.0042938	-0.00713438	0.00832683

Figure 4.9: An example of data output from the A*steroid Game*, showing current target and cursor positions, and how much time has passed in milliseconds

gaming system to output high quality data. To facilitate this, the framework provides a modular approach to data output, whereby a number of metrics, both those internal to the framework's core, and those related to a specific game built using the framework, can be chosen to be output for every frame that the game executes. At the start of program execution, class member variables to be output to file are registered with an output handling class, with a name, data type, and pointer to the variable in memory. The contents of this memory can then be snapshotted, formatted as appropriate (such as enforcing the number of digits after the decimal point in float datatypes), and written to a text file by the output class. The output class supports outputting strings, boolean, integer, and floating point data; of the data types, only string cannot be natively assumed to be wholly updated atomically, and is therefore only used to record data set once throughout program execution, such as the player's name.

4.8.1 Format

All selected data metrics are outputted in a standard separated value format, whereby data entries are separated by the pipe ('|') symbol. Once all required data has been registered with the system, data output can commence to a file with a specified name and path. The resulting text file will consist of a number of columns of data; each column represents one variable, with each row being the data sample for an individual frame. The first data point of each column will be the text name provided to the output class on variable registration. This results in output data formatted as in Figure 4.9; note that the boolean datatype outputs 1 for true and 0 for false.

4.8.2 Representation Of Device Data

The framework provides access to a variety of metrics relating to the current state of the force sensors it is receiving user input from. The current normalised value of each sensor is available for logging, as is the current maximum calibrated force discussed in Section 4.4.1, and any offset added to compensate for issues outlined in Section 4.4.3.

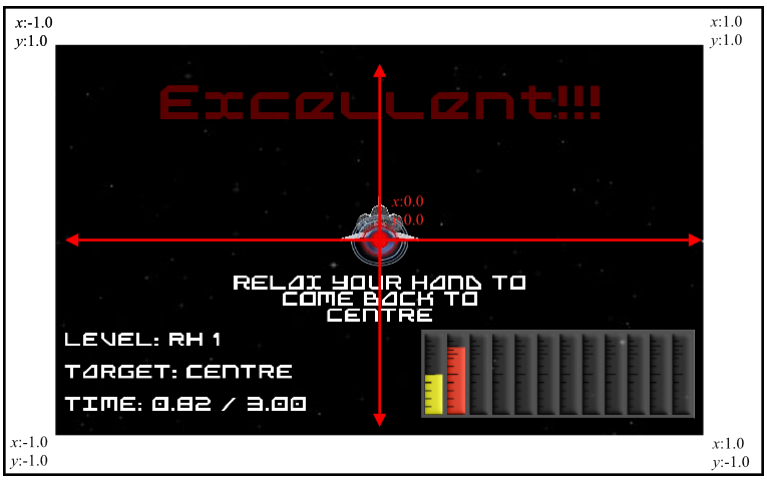


Figure 4.10: Spatial data is recorded to data files in a normalised space, running from -1.0 to 1.0 on each axis

4.8.3 Representation Of Spatial Data

For analysis of a player's performance in game, it may be necessary to output the position of something on screen, for example to calculate how close a player character is to an enemy. Regardless of screen resolution, positional data relating to the screen is output in a normalised format, whereby data on each axis has extents of -1.0 to 1.0. This matches the 'normalised device coordinate' space used in computer graphics literature. An example of this is provided in Figure 4.10.

4.8.4 Frequency Of Data Updates

The requested data metrics are polled at the end of each frame to obtain their latest contents and output them to file. It is usually the case in a video game for the number of frames calculated each second to be locked to the *vertical sync* of the monitor displaying it - that is, how often the raster beam is reset to the top line of phosphorous elements (for CRT monitors), or an internal memory buffer is copied upon receiving a signal (for LCD monitors). Most display devices utilise a vertical sync rate of 60hz, though it is common for games to only update their graphical output at a rate of 30hz (once every other sync signal) so as to afford greater time in which to render their increasingly complex scenes. For the purposes of the trials undertaken using the framework outlined in this chapter, this v-sync value was ignored, and the internal logic of the games was instead refreshed internally at a rate of 500hz. This is advantageous as it precludes the loss of high-frequency data from the user's input - rapid oscillations or changes in user pressure could be indicators of fatigue, loss of concentration, or some other phenomenon that is easier to infer with a higher fidelity of data. This rate is at least twice that of any of the input devices used, ensuring that no detectable user input is missed, as per the Nyquist Sampling Theorem.

Figure 4.11: A gameplay shot of the Asteroid Game: The player must accurately place the red cursor over the blue target using the force sensors, as the target moves around on screen.

4.9 The Asteroid Game

To aid in the search for the response markers that could determine whether a particular rehabilitation regime is effectively reaching its desired outcomes, a simple rehabilitative game was built upon the framework, and used in a number of user trials of able-bodied and post-stroke participants, as part of wider research undertaken in conjunction with the Institute of Neuroscience at Newcastle University. The core game features were kept simple so as to be modified as necessary over the course of a number of test trials, as trial data was examined and new potential research avenues present themselves, both in terms of direct evidence of response markers, and of evidence of interlimb transfer of motor function.

4.9.1 Description of Asteroid Game

In gameplay terms, the goal of the example game is to use the force-sensing input devices to move a cursor around on screen using the control schemes outlined in 4.9.4 and devices outlined in 4.5, with the gameplay goal of placing the cursor quickly and accurately over a series of on-screen targets, presented to the user sequentially. This basic gameplay mechanism is presented thematically as the user controlling the targeting mechanisms of a space ship, which must lock on to, and shoot down, a number of asteroids flying through space.

Cursor position is directly proportional to the player's current applied force, giving the appearance of smooth movement across the screen as the force applied by their hand is ramped up and released. Care should be taken when designing a game with such emphasis on input analysis not to cause erroneous movements and confu-

sion by attempting to inject aspects of realism into the gameplay: a vehicle may not really display such inertia-less instantaneous movement, but realistic applications of acceleration and velocity to movements add another game mechanic for the user to be confused and distracted by [ALMK10b], with little rehabilitative benefit.

Central to the game is the concept of a *training* hand, and a *non-training* (or *receiving*) hand; these represent the able, and plegic limbs, respectively. Being able to successfully replicate complex actions with both limbs is an important indicator of a return in dexterity to a hemiplegic limb, and is a possible indicator of inter-limb transfer of motor function, which was of particular interest to the research collaborators. Throughout the game execution, the training and non-training hands are treated separately, with separate calibrations maintained for each as in section 4.4.2; this allows the performance of each limb to be tested and monitored within its own level of strength and dexterity.

Progress through the game is divided into a number of *stages*, with each stage consisting of a number of *trials*, to be completed by operating the on-screen cursor with either the training or receiving hand. Each of these trials is a gameplay segment consisting of up to 12 targets, with the player being scored based on how quickly, and how accurately, they placed the cursor over each target using the specified hand. At the end of every trial, an overview of the player's performance is presented, and instructions as to which hand to ready for the next trial shown. This continues until a defined number of trials have been played, whereon the game will display a final score on screen, and output a number of text files containing score information for later analysis, before exiting the program. The number of stages, trials, and targets within each trial is all modifiable using the gameflow file mechanism discussed in Section 4.7.2.3.

4.9.2 Target Action Phases

Each of the targets within a trial may appear on screen in a number of different positions. In order to successfully 'destroy' the target, the user must complete three linked sequential movements: *resting*, *locking*, and *holding* (see Figure 4.12).

Resting: The user is first instructed to centre their cursor at the middle of the screen, by resting their hand completely, with as little muscle contraction as possible. This allows each target attempt to be analysed in a uniform, discrete manner - were this first step skipped, the relative ordering of on-screen target positions in the trial could have an undesirable additional influence on performance, skewing results. During this stage, the input handling functionality of the framework performs a recalibration of the active device's parameters, as described in section 4.4.3.

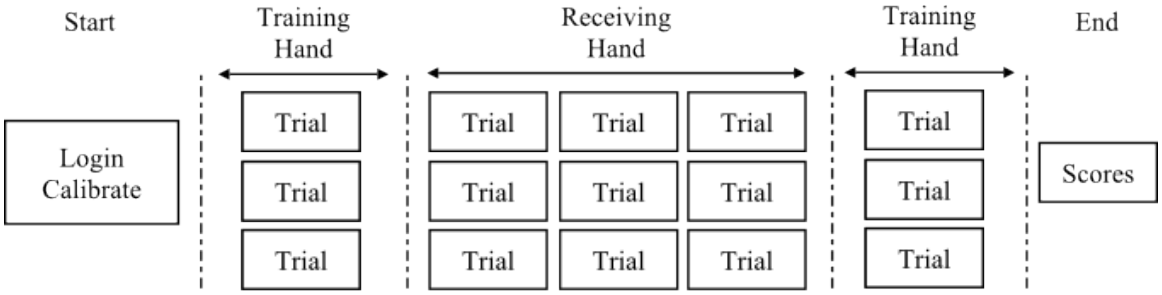
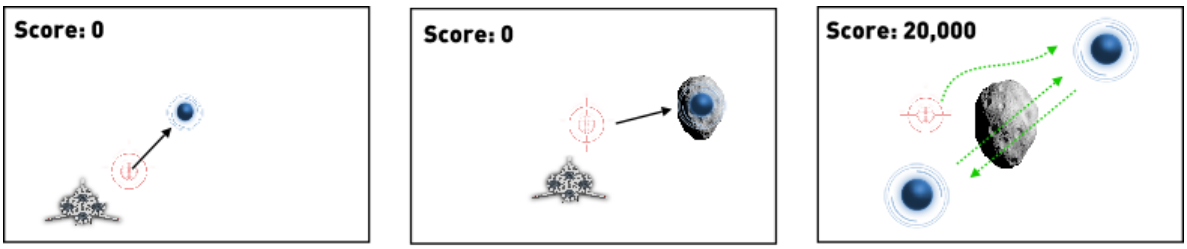


Figure 4.12: Default setup of trials and phases: The user performs 3 trials of 12 targets using their training hand to determine baseline ability, followed by a longer period with their receiving hand, with a final 3 trial stage to determine end ability with the training hand.

Locking: Upon completion of the resting phase, the target is presented on screen, and the player must perform a further two actions: the initial 'locking on' of moving the cursor over the target, and 'holding' the cursor over the target for a period of time. This combination of tasks is important: a goal of rehabilitation is to allow a patient to regain or strengthen the ability to perform a variety of 'daily living' tasks, such as buttoning a shirt or opening a door, many of which are compound actions, requiring accuracy and coordination between movements to complete. Therefore, the ability to learn and improve on such combination tasks in the context of a rehabilitative game could be an important response marker.

Tracking: When a target first appears, it will remain stationary until it has been 'locked on' to by hovering the cursor over it for a predetermined amount of time, at which point any movements that have been programmed for the target will begin. The player must then hover their target over the (potentially moving) target, until an on-screen timer expires, upon which the target is considered 'destroyed', and the next iteration begins. If a player fails to successfully lock on or destroy a target, for whatever reason (such as inaccuracy, or misunderstanding the instructions on screen), an internal timer (of a user-configurable length) will eventually expire, whereon the game will advance to the next state; either treating the target as locked-on or destroyed as necessary. This prevents a player from becoming permanently 'stuck' within a section of the game. Details pertaining to each of the targets within a phase are determined via nodes within the loaded gameflow file, described in more detail in Section 4.9.6.1. Each target can appear at a unique position on screen, and may additionally move on screen to present an extra hand-eye coordination challenge to the player.

These stages were designed in collaboration with researchers within the Institute of Neuroscience so as to meet multiple research goals; while investigating the capability for motor task skill learning over time to be measured, the idea of investigating



Rest Phase: Lift Off Sensor, Cursor moves to centre of screen

Locking Phase: Target appears on screen. Push on sensors to move and hold cursor over target

Holding Phase: Target may move in a pattern. Adjust force on sensors to keep cursor over target

Figure 4.13: Asteroid Game Target Phases: The initial rest phase is used to recalibrate the sensors, and to reduce any scoring bias from the previous target's position. The locking phase requires the user to push the sensors to quickly move the cursor over the target. The final holding phase requires the player to follow the moving target as accurately as possible.

a link between methods of task instruction and long-term performance necessitated tasks that had multiple, intuitive stages that could be analysed separately.

4.9.3 Scoring Metric

Game players receive a score for each target, derived from both the speed at which they 'locked on' to the target by moving their cursor over it, and their accuracy at staying over the target. In this way, a combination of well controlled reflexes and muscle contraction will result in higher scores - higher scores will therefore equate to better overall control over the limb. The time taken to 'lock on', and the mean distance of the player's cursor from the target are logged to the output file as play progresses.

The score produced for each target is a product of two performance metrics, designed to provide an indicator as to the player's hand-eye coordination, and their ability to successfully predict and plan a combined muscle movement. This takes the form of two sub calculations for speed and accuracy, weighted equally:

$$score = 100(\frac{speed + accuracy}{2}) \tag{4.2}$$

Both sub calculations utilise the mean Euclidean distance between the player's cursor and the target, measured in normalised coordinate space:

$$mean = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2} \tag{4.3}$$

While the accuracy calculation also requires the mean Euclidean distance between the cursor and the target over time:

$$mean = \frac{\sum_{i=1}^{n} \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}}{n} \tag{4.4}$$

Where a and b are the cursor and target positions, and n is the current number of samples taken - as this score is calculated every frame, it can change over time, as the number of samples n will increase.

The speed score is determined by the time taken between a target first appearing on screen, and the player's cursor being placed over the target for a total of 0.5 seconds; this is known as 'locking on' to the target, from the in-game parlance used to describe game progress. A time threshold rather than a single distance check is used in order to reduce occurrences of 'incorrect' scoring, whereby a player moves their cursor through the target without stopping to complete the 'holding' action - this is of particular concern when dealing with patients with disabilities, where the weakness and lack of control in the player's limb could present as shaky movement [HMHD09] more likely to move in and out of a target area, resulting in 'false' readings.

A target distance of less than or equal to the radius of the target in normalised coordinates is considered a 'hit', with every game frame in a hit state counting towards the 0.5 second timer. The total time taken between the target appearing and the 0.5 second timer being depleted is then compared against reference minimum and maximum reference values loaded in the configuration file, resulting in the following speed score calculation:

$$speed = sin(90(\frac{\alpha - minTime}{maxTime - minTime})) \tag{4.5}$$

In this equation, α is the time taken to hover over the target for 0.5 seconds, *minTime* is the minimum time by which the user can score 100%, and *maxTime* is the time at which the player will score 0% for the speed metric.

Accuracy is scored similarly:

$$accuracy = sin(90(\frac{\beta - minDist}{maxDist - minDist})) \tag{4.6}$$

Where β is the mean distance from the target recorded since the target was locked on to as in Equation 4.4, and *minDist* and *minDist* being a minimum and maximum distance allowed from the target - these values are more fully detailed in Section 4.9.7.

4.9.4 User Input

Within the game, the user is required to move a cursor on screen, using either their training or non-training hand, determined from a per-target variable within the gameflow file. As part of the calibration procedure enacted at the start of program execution, the *mean maximum pressure* the player is capable of enacting upon each of the sensors is calculated. For the Asteroid Game it was decided to further scale the input magnitude such that the player must use no more than 40% of their maximum pressure to hit a target in its starting position - the customisable movement of each target may require more pressure to track it.

As a player is expected to play a series of trials over a time period of half an hour or more (the exact time being determined by how quickly they successfully destroy each target), such accommodations must be made to ensure that the player does not become fatigued or suffer discomfort throughout their gameplay experience, negatively impacting performance and the medical relevance of their results. The 40% of maximum pressure limitation was a compromise; while works such as Massy-Westropp et al. [MWRA$^+$04] note that the average person can maintain an enacted pressure of between 39% and 63% of their peak grip strength over short periods of time, the desire to design an experiment that could be undertaken by impaired individuals without causing excessive fatigue led to the decision to utilise a low value for the desired force with the experiment. In order to make best use of the data being collected from the sensors, it was decided to not use a value *too* low, however, to minimise the effect of signal noise upon the recordings - tests utilising a maximum value of 10% led to signal noise causing noticeable oscillations in cursor position, and excessive quantisation of the user input due to 'throwing away' such a large amount of the base data due to using such a small amount of the various device's detection sensitivity.

The means by which the user's physical movement is transformed into on-screen movement of the cursor differs slightly depending on whether the game is being played with a device with 2 orthogonal axis of movement (such as the Saitek joystick), and those utilising pairs of 1 dimensional force sensors (such as the iLoad Mini and C500). Ensuring that each device was as intuitive as possible to use was seen as high priority, with works such as Ryan et al. [RRP06] and Alankus et al. [ALMK10a] noting the link between gameplay enjoyment and intuitive, readily mastered input.

4.9.4.1 Two Axis Movement

For force sensors with 2 fully separable axis of movement, the mapping between device force and on screen cursor movement is simple. For the x-axis, the extents of

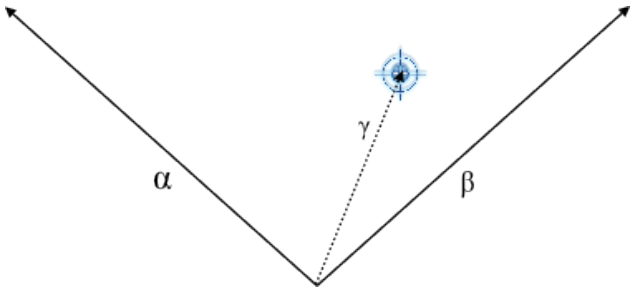


Figure 4.14: The hand sensors are used to place a cursor somewhere in the arc between v_0 and v_1, based on the total amount of force applied, and the ratio between their relative applied forces.

calibrated movement map directly to the screen-space extents, while for the *y*-axis, the extent of positive force (that is, when pushing on the joystick away from the body) maps to the top of the screen, while force in the negative direction (pulling the device towards the body) maps to the centre of the screen, resulting in the lower half of the screen being impossible to enter.

4.9.4.2 Two Device Movement

For devices consisting of multiple individual sensors, a mapping must be made between the values reported from the sensors placed under the hand, and the 2D space the cursor must move in. To allow this, two normalised 2D direction vectors (denoted v_0 and v_1) define a 'cone' in screen-space within which the cursor can move; ideally these vectors will mirrors of each other from the centre of the screen. Using these in conjunction with the magnitude of the input of the two sensors under the hand, a further vector γ can be formed (see Figure 4.14). Vector γ is simply calculated as:

$$\gamma = v_0\alpha + v_1\beta \tag{4.7}$$

Where α and β are the normalised magnitudes of the force sensor placed on the left and right, respectively. The resulting vector γ can then be used as an offset from the centre of screen-space to position the user's cursor on screen.

4.9.5 User Considerations

As the Asteroid Game was to be used by people who may not necessarily be familiar with video games, or with novel methods of user input like the force sensing devices, it was seen as important to reassure the player that they were doing well. Works such as Chen [Che07] note that the ideal gameplay experience is one that is neither too easy so as to promote boredom, nor difficult enough to cause anxiety, but to instead offer the player a mixed gameplay experience that they can enjoy,

promoting a feeling of happiness and fulfilment often known as 'flow', or being 'in the zone'.

To promote this desirable feeling of game 'flow' within the Asteroid Game, it was seen as important to provide the player with the perception that they were doing well, even if by the strict measures of their in-game performance they were not. This manifested itself in several ways in the design of the game. For example, upon completion of a target, the user is informed via on-screen textual feedback as to their performance, ranging from 'OK', to 'Good' and 'Excellent'. In this case, even a performance that was in actuality very bad will elicit an 'OK' result, so as not to make the user feel they are performing badly, which may result in not engaging with the game. Similarly the use of the *sin* function when calculating the scores discussed in 4.9.3 was to change the linear score degradation from 100% to 0% to instead have the shape of a circle quadrant, pushing middling scores upward, and allowing a relatively poor target attempt to still achieve a perceptually 'good' score, limiting negative side effects from giving a low score to the user. During gameplay itself, the user is kept aware of their scores for the current trial via a chart in the bottom left corner (see Figure 4.11); this was added to provide players with an achievement to aim towards in beating their previous trial result, affording some degree of personal reward to the gameplay experience.

4.9.6 Configurable features

The particulars of the game, such as number of trials, targets per trial, time until target destruction, and even seemingly minor details such as the length of time an instructional message appears on screen, have been made fully customisable via state definitions within the gameflow file, in the format outlined in Section 4.7.2.3. By allowing easy modification of the game without requiring programming knowledge or recompilation of the program, it becomes easy to rapidly build variations of the game to focus on a particular area of research interest. This has proved to be invaluable during the course of user trials utilising the software; for instance therapists initially found the length of the game to be too arduous for some users, and the chosen movement parameters too difficult for test subjects with extreme disability, problems easily solved via modification of a text file, without requiring software development tools.

4.9.6.1 Target Logic

Each target in a trial is defined by three parameters: its starting position, its movement logic, and the time it takes to be 'destroyed' by holding the cursor over it. By default, each target will take these parameters from the currently loaded game-

flow, which is assumed by the game to have a number of child data sets of type *Target*, consisting of keys representing the starting position on screen, the time in milliseconds to be destroyed, and a numerical ID which will be written to output files. There is also an optional *Random* tag, a boolean value which if true, means that a target obtains its position and destruction time from another data set, of type *RandomData*; this contains a number of position and destruction times, which will be chosen from randomly whenever that target is reached - therefore, Target 0 in a number of different trials can have different parameters if the *Random* tag is set. It is possible for an individual trial to override the default target data set in a gameflow simply by defining its own data sets of type *Target*: a Target data set with the name *Target0* will override the default logic of the first target in a trial, with all other targets being sourced from the gameflow.

Each target may either be stationary, or have a number of simple movement logic rules dictating its position over time. These rules, known as *modifiers*, are shown in Figure 4.17, and define a movement along an axis at a velocity defined by either a sine or a cosine wave. Each wave is configured using variables for both amplitude and frequency, and each axis can be defined as being either the *x*,*y*, or *directional* axis; the directional axis being the normalised direction vector between the centre of the screen and the target starting point. These parameters afford a great deal of variance in movement of a target over time: a directional axis oscillation will make the target move closer to and further away from the centre of the screen, while a combination of two modifiers using a sine and cosine on the *x* and *y* axis will make the target move in a circular motion.

For the purposes of the trials the Asteroid Game was used in, it was decided not to calibrate the velocity of the modified targets, so as to provide as close to the same gameplay environment as possible to each trial participant. In a scenario where the Asteroid Game were to be used in the home, it would however be beneficial to scale the velocity of targets to user ability; it may be necessary to 'calibrate' the target velocity not just by the mean force imparted by the user upon the input devices in the initial calibration stage, but also on how quickly tasks have been performed previously.

4.9.6.2 Game States

The Asteroid Game employs a number of game state classes, which can be ordered or omitted as necessary using the *gameflow* system outlined in 4.7.2.3; the only restriction being that if present, the *login* state must come first, and be present within the gameflow only once, to avoid string extraction issues when writing output to file, as discussed in 4.8. The provided states include the following:

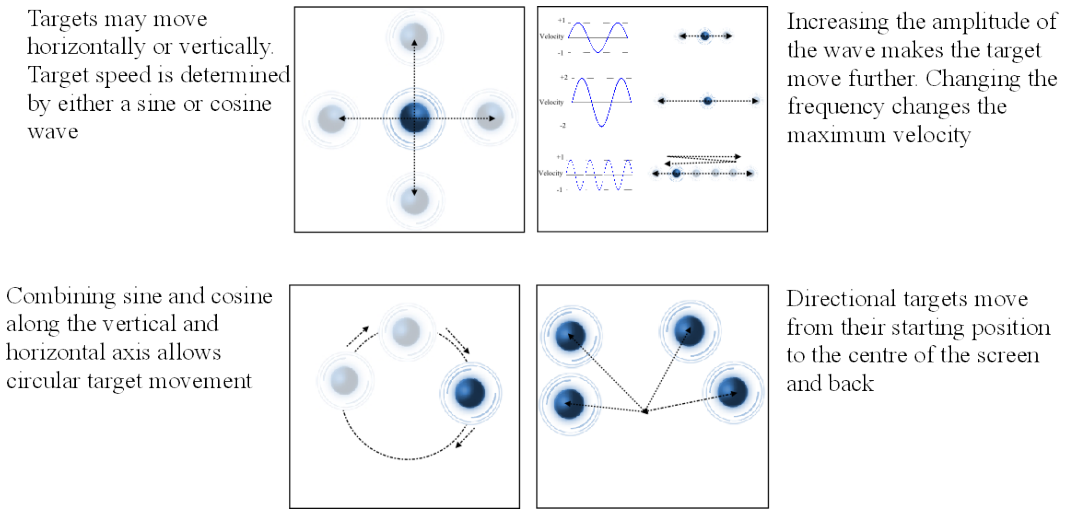


Figure 4.15: Overview of Target Modifiers: Targets can move horizontally, vertically, and towards the centre of the screen, at a selectable frequency (how quickly it moves across the axis) and amplitude (how much it moves on the axis). Modifiers can be stacked to create more erratic movements.

- **WaitMessage**: Message informing user to rest and wait.
- **IncrementBlock**: Increments or resets a counter, used in data output.
- **ZeroSensors**: Message informing user to place hands on sensors, to tare.
- **SwitchJoystickHand**: Switches game to use either training and receiving hand
- **AsteroidGame**: A trial consisting of up to 12 targets
- **AsteroidPractise**: As above, but calculates player metrics on completion

The *IncrementBlock* and *SwitchJoystickHand* states immediately pop themselves off the state stack after a single frame, after performing their required actions. States like these were implemented so as to allow easier setting up of custom gameplay parameters - the *SwitchJoystickHand* state can be placed between trials or targets, and will change and inform the player of which hand will be required for the next gameplay stage, while the *IncrementBlock* state increments an internal counter, which is logged to the output file every frame as an easy way of identifying grouped trials.

4.9.7 Practice Sessions

The size of the targets on screen, as well as the minimum and maximum score distances used in the score Equation 4.3 can be adjusted based on user performance. This is achieved by determining the player's ability to play the game with an initial practice session, to be undertaken after the force sensors have been calibrated, but before the main portion of the game has begun.

The gameflow node that initiates the practice session is assumed to have three arrays of 10 values, that set the target size, minimum scoring distance, and maximum scoring distance, respectively (shown in Figure 4.16). All values are measured in the normalised units discussed in Section 4.8.3. The minimum and maximum scoring ranges may be larger than that of the target graphic on screen if desired, allowing the player to be scored for 'near misses' if desired. The practice session itself consists of a single in-game trial of between 1 and 12 targets. As with other game trials, the positions, modifiers, and hand to use for each target can be set by the gameflow, with all targets being set to the parameters at location 5 of the 0-indexed array of target size information.

Upon completion of the practice session, the mean score achieved is determined, and quantised into one of 10 values - a score of 54% will be quantised to 5, while a score of 94% will become slot 9. The resulting value is then used as an index into the three target size information arrays for the rest of the duration of the game play. This allows for a degree of adaptability towards players of differing skill levels - those who are good at the game can be presented with a smaller target, and thus a harder challenge to complete, while those who are yet to master the game are presented with an easier gameplay experience to encourage continued efforts to improve. The practice session is optional, and not including it in the gameflow will case the game to instead use default values.

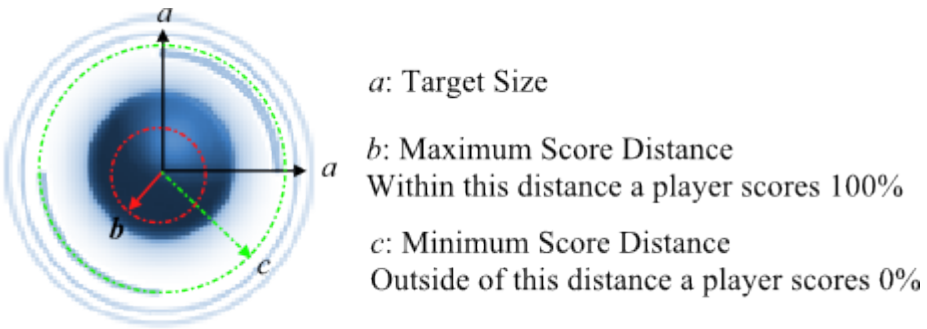


Figure 4.16: Overview of Target Scoring: The user's cursor being within the inner red circle will yield high scores, while being outside of the green circle will negatively impact scores. Both distances can be set within the Gameflow file.

4.10 User Trials

The Asteroid Game built using the game framework outlined in this chapter has been successfully used in user trials, as part of a wider research effort in conjunction with Prof. Janet Eyre's team at the Institute of Neuroscience at Newcastle University, with all results derived in collaboration with them.

These trials have been used to investigate a number of different aspects of video game delivered rehabilitation, including links between in-game motor control performance and real-world increases in motor ability, and whether video game technology is sensitive enough to detect whether motor learning of sequential actions is taking place over time. Demonstrating the ability to determine whether or not motor learning is taking place or not could be used as an indicator of the overall condition of the patient's stroke related symptoms over time, and demonstrate the viability of video games as a tool for therapeutic benefit.

A second desired outcome for the research was to determine whether the way an in-game motor control task is presented alters the user's ability to improve their performance at that task over time. If this were to be shown, and additionally combined with a link between in-game performance and real-world ability, then the research could enable more effective and efficient means of creating interactive rehabilitation software. This research goal was tested via analysing the performance of two groups of players, with each group informed of their in-game task in a consistently different manner. The metric of performance for this derived from the data used to calculate the scores within the Asteroid Game, as discussed in Section 4.9.3.

The framework was designed to easily facilitate such research outcomes, by allowing for high quality data from the motion sensors and game state to be logged to file, and rapid adaptation of the game as necessary in order to extract as much useful data as possible over the course of the research trials. The interdisciplinary nature of the research team meant that the people utilising and modifying the Asteroid Game via the gameflow mechanism were not necessarily proficient at computer programming, making ease of use vital for its continued use and success.

4.10.1 Trial Team

The work described in this thesis chapter was part of a larger collaboration between the School of Computing Science and Institute of Neuroscience, investigating methods by which interactive media such as computer games could increase and add a sense of reward to physical rehabilitation routines. This involved a number of projects involving video game technology, including software maintenance and technical assistance on trials being performed with Circus Challenge, analysis of patterns within 9-hole peg board play attempts, and the project which resulted in the creation of the Asteroid Game, which was initially to investigate whether physical movements delivered via video games and attempted with a user's non-plegic limbs would result in improvements in coordination with their plegic limb, but expanded as trials were undertaken.

Due to the interdisciplinary nature of the work, a number of different people were involved throughout the course of the trial. The author handled all of the data generation via the creation of the game engine, the Asteroid Game itself, and the data output and gameflow. In addition to this, several members of the Institute of Neuroscience were involved:

- **Janet Eyre**: Janet is Professor of Paediatric Neuroscience at Newcastle University, and headed the team at the Institute of Neuroscience, coordinating efforts between research groups.
- **Kholood Shalabi**: Kholood is a PhD student supervised by Prof. Janet Eyre, investigating the links between hemiplegic motor learning and health gaming. Kholood's role in the work described in this chapter was to conduct the actual trials themselves; she organised trial sessions, explained the process to participants, and ensured that each session was completed successfully.
- **Sara Graziadio**: Sara is a post-doctoral researcher within the Institute of Neuroscience. She conceived the idea of investigating the potential for inter-limb transfer of motor function, and designed the initial concept of what became the Asteroid Game. Beyond this, Sara also performed statistical analysis on the results of the Asteroid Game, along with Kholood.
- **Occupational Therapists**: A number of occupational therapists were attached to the larger video game therapy project as a whole during the course of the Asteroid Game trials, and their feedback and experience of working with patients during physical therapy was useful in designing the trials to ensure that they were not too strenuous for impaired persons to complete.

4.10.2 Experiment Data Output

For the duration of the research trials, the framework was configured to output the following data output for later analysis, with the latest values written to text file at the end of every game frame:

- **Time** (*float*): Time in milliseconds since data capture began
- **Block** (*int*): Current stage of the experiment
- **Trial** (*int*): How many trials into the block is the experiment
- **Target** (*int*): Index of current gameplay target
- **Fixed / Rand** (*bool*): Whether the current target properties are random
- **State** (*char*): Identifier for the current state of the game

- **CursorX** (*float*): Normalised *x* axis position of the player cursor
- **CursorY** (*float*): Normalised *y* axis position of the player cursor
- **TargetX** (*float*): Normalised *x* axis position of the current target
- **TargetY** (*float*): Normalised *y* axis position of the current target
- **TargetDist** (*float*): Normalised distance between player cursor and target

For each of the force sensors, the following is also recorded:

- **Raw** (*int*): Raw value reported by the driver
- **Min** (*int*): Minimum value reported
- **Max** (*int*): Maximum value reported
- **PSI** (*float*): Force in PSI applied to the sensor

The state tag uses a number of predefined values within the game to tag game frames in which the internal workings of the game changed in some way; for instance, new tags are written to the output file on the frame in which a new target appears, is 'locked on' to, or is destroyed, and whenever the sensors have their tare weight calibrated. This state tagging, together with the full output of sensor values, provide a complete state of the game at all points throughout its execution.

4.10.3 Experiment Protocol

For the user trials, a strict protocol was devised, which would allow for consistency over a number of test sessions for each participant. Much of the protocol was designed by the Institute of Neuroscience staff outlined earlier, but several suggestions and modifications were made by the author: presenting moving targets was a suggestion from myself; also, as noted in Section 4.9.4, initial trials were initially to utilise only a maximum of 10% of a user's mean pressure, but after discussion a compromise of 40% was utilised, to allow for greater use of input device sensitivity, balanced against the need to not fatigue the user over the course of the trial.

The ethical committee of Newcastle University approved the trial study, and written informed consent was obtained from all participants. Although some amount of user data in the form of force applied to sensors ever frame, along with the user's age and whether they were left or right handed, this was anonymised via a simple identification number used for referencing recording files.

Each group played an identical version of the Asteroid Game, set up using the gameflow file described throughout this chapter. This setup divided the game into

Trial Type	Receiving	Training	Training	Training	Receiving
Trial Count	3	3	15	3	3

Table 4.1: Experiment Trial setup: Number of trials performed with each hand, presented to the player as ordered from left to right.

the three stages outlined in Section 4.9.2. The first and last stages were further modified so as to include trials for both receiving and training hands, resulting in the trial setup outlined in Table 4.1. Each experiment participant's play-through of the game was standardised to reduce external influences in player performance, with the same computer and sensor setup used, and a strict script which the trial supervisor kept to when communicating to the trial participant what they should do during the play session.

Before any user trials were started, the Asteroid Game underwent an initial test with members of the Institute of Neuroscience, and students from the School of Computing Science at Newcastle University. This was primarily to test the stability of the program for extended periods, and ensure that the data recording was correct. As part of this test, it was determined that the original length of the game session, consisting of 20 trials in the second stage, was too long, with participants stating that they were starting to tire of playing the game. Due to this, the number of trials was reduced, resulting in the trial setup noted in Table 4.1. Based on feedback from these initial tests, the target colour was also changed when 'hitting' the target, and the speed at which the 3D spaceship model moved adjusted, as people were sometimes confused as to whether they were moving the spaceship or not, as it would initially interpolate towards the cursor position over time; changing the speed to instantaneous made the game more intuitive.

The first two trial sets serve as a baseline measurement for player performance using each limb. The longer trial performed using the training hand is used as a practice and learning period. The final training hand trial set allows for an overall measurement of the player's ability to learn the task presented to them, while the final receiving hand trial set allows for a measurement of potential skill transfer from the strong to weak limb. Participants were informed of their progress via on-screen display of their scores, as defined in 4.9.3. For analysis, participant performance was measured from the two metrics used to create that score - the time taken to 'lock on' to the target (also known as the target's *Transfer Time*), and the mean distance between the user's cursor and the target. The distance metric was recorded directly each frame as part of the data output, while Transfer Time was obtained by measuring the distance between 'New Target' and 'Locked On' State symbols within the data output - as each row is timestamped, it is trivial to determine the time between state changes via subtraction.

Group	Mean Age	Minimum Age	Maximum Age
1	27	20	35
2	27	20	36

Table 4.2: First trial age ranges per group, measured in years.

	Group	Time	Transfer Time	Distance
Group		0.015	0.256	0.012
Time			0.001	0.001

Table 4.3: ANOVA Effect Statistics: Training Hand

4.10.4 Participant Grouping

For the purposes of this trial, it was decided that as it was designed around how to test how a user's perception of a task affects performance over time, rather than the game's specific ability to aid in post-stroke rehabilitation, trial participants had no requirement to have suffered stroke. Therefore, participants were sourced from able-bodied volunteers. Trial participants were divided into two groups, consisting of 12 right-handed adults evenly split between gender, with age ranges as shown in Table 4.2, with each group being tutored in how to play the game in a different manner, altering their perception as to how to achieve high scores. Group 1 was tutored in how to play the game as if success was achieved with a single objective - they were asked to simply track the target asteroids as best they could. Group 2 was tutored tutored in gameplay as a two stage process - they were asked to first move their cursor over the target as quickly as possible, and to then track the target as accurately as possible. Trial group 2 therefore had the concept of multiple steps being required for success introduced to them early-on, whilst group 1 would have to infer this manually from game scoring feedback during play.

4.10.5 Experiment Outcomes

The main results of the trial are collated in Figure 4.17. The graph labelled *Transfer Time* shows the mean time in seconds taken for each group to achieve the 'lock on' state for a target. The *Distance* graph shows the mean distance between the user's cursor and the target, measured in the normalised screen space outlined in Section 4.8.3. Lower values in the y-axis are 'better' scores in both cases. In both graphs each data point is a mean value taken from 3 trials of 12 targets, with blue points indicating data from Group 1, and green indicating Group 2.

	Group	Time	Transfer Time	Distance
Group		0.04	0.034	0.001
Time			0.001	0.001

Table 4.4: ANOVA Effect Statistics: Non-Trained Hand

In depth analysis was performed on the trial data using the SPSS software package by IBM (www.ibm.com/analytics/us/en/technology/spss/; *IBM*). The General Linear Model Repeated Measures Analysis of Variance statistical model was used, with Greenhouse-Geisser correction used where necessary. All statistical results used a significance level of 0.05, with Bonferroni correction. Each hand (either the trained hand or non-trained hand) was analysed separately. An overview of this analysis is provided in Tables 4.3 and 4.4. There are two immediately obvious observations to be made from inspection of the graphs: that both groups plateau in their performance over time, and that Group 2 is consistently worse across both transfer time and distance.

Analysis proved a main effect of *Group* for the *Distance* metric ($p = 0.012$ trained hand, $p = 0.001$ non-trained hand), demonstrating a significant advantage to introducing the task as a single combined motor task. Both training and non-training hands showed a main effect of *time* for transfer time and distance ($p < 0.001$ in all cases), indicating that over the course of a participants trials, motor learning was taking place. The effect of time and distance even for the lesser-used non-trained hand is an indicator that there was some degree of inter-limb transfer of movement training from the usage of the other limb. This effect was present for both groups. The effect of transfer time over time was not significant for the training hand ($p = 0.256$), but significant for the non-trained hand ($p = 0.034$), additionally showing significantly better performance by the single object group.

4.10.6 Second User Trial

As part of ongoing research using the Asteroid Game by researchers within the Institute of Neuroscience, trials have been undertaken with older people who have previously suffered stroke. The primary purpose of this particular trial was to determine whether the game could be used to measure the learning of sequentially linked actions in those suffering from hemiplegic stroke. Preliminary results are discussed here, as further evidence as to the ability of the Asteroid Game to assist in determining how video game technology can assist in stroke rehabilitation; work is currently being undertaken to expand upon these results with the aim of submission to the *UK Stroke Forum* conference.

4.10.6.1 Second Trial Setup

The second trial is similar in structure to that of the first, consisting of groups of people playing the Asteroid Game using the same configurable gameflow and data output, allowing comparisons between game sessions to be made. Table 4.5 outlines

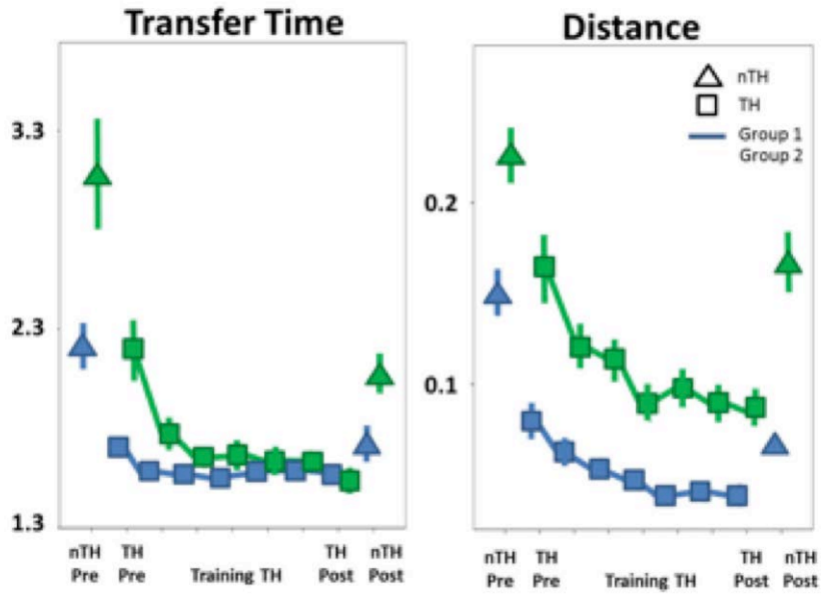


Figure 4.17: Graphs derived from game data; learning curves for transfer time (s) and distance. For analysis the datafor every three sequential trials are grouped. The blue symbols indicate Group 1 (perceived single objective) and the green symbols Group 2 (perceived 2 step task). The squares indicate the trained right hand (TH) and the triangles the non-trained left hand (nTH). The error bars are SEMs. For both indices lower values are associated with higher performances

how the trial participants were grouped. Unlike in the previous trial's participant mean age of 27 years of age, this trial focussed on older people, with an average age of 65 years. Of those participants who had previously suffered from stroke, all were at least 6 months post-stroke. All trial participants were right-handed, but were further split up into groups where either the left or right hand was considered the 'training' hand with which the game was primarily played with; stroke group participants trained their paretic hand, and were grouped up as such.

The game's gameflow setup file was similar to that of the first research trial, with 12 targets in each individual game trial, and 12 game trials making up a single play session; two sessions are undertaken by each participant, 7 days apart, in order to allow a measure of longer-term motor learning to be analysed. Targets were presented to the player in one of four randomly selected positions, with target modifiers setup to slowly move either horizontally, vertically, or towards the centre of the screen and back.

4.10.6.2 Second Trial Results

Analysis of the trial data was performed as before, with a General Linear Model Repeated Measures Analysis of Variance statistical model. Within subject *time* (times taken to lock on to and successfully track a target to its destruction) was analysed,

Group	Count	Mean Age	Age Range
Able Right	13	66.2	$\pm$7.1
Able Left	14	65	$\pm$9.9
Stroke Right	6	62.7	$\pm$7.6
Stroke Left	10	69.8	$\pm$9.8

Table 4.5: Second trial age ranges per group, measured in years.

while between subject analysis was in *group* (whether able bodied or stroke survivor), and in training hand (right or left). The analysis of performance within the trials showed a significant main effect for *time* ($p = 0.011$ for locking, $p < 0.001$ for tracking), indicating learning was occurring over time. A main effect was also determined for grouping in regards to ability to lock and track targets, with a worse performance for stroke survivors ($p < 0.001$). For the locking phase of the target, a trend towards *time/training/group* interaction was noted ($p = 0.092$), with stroke survivors with left-hemiparesis showing no learning over time. In tracking, superior right hand performance was indicated, via a main effect for the *training hand* ($p = 0.028$). A *group/training hand* interaction was noted ($p = 0.036$), with a greater difference in performance between hands for the stroke and able bodied groups.

4.10.6.3 Second Experiment Outcomes

The second experiment with stroke survivors shows some results which, while mostly unsurprising, are nevertheless promising for having been determined solely from the data output during played of the Asteroid Game. The significant effect for group across the locking on and tracking targets indicates that that in-game performance is impaired in those suffering from hemiplegic stroke compared to the control group. Nevertheless, learning was indicated by means of a significant main effect for time. The ability to lock a target was noted to be slower for the impaired groups, with non-dominant hemisphere stroke participants showing more significant impairment in tracking and learning, evidenced by a time/training/group trend, and a a significant main effect for handedness. The interaction between group and training hand, and the greater difference in performance between hands and groups, indicates that those participants who are both post-stroke and have hemiparisis in their dominant hand are likely to perform worse; that despite this the results indicated learning over time is promising, as it would mean that enough sensitivity was present to detect changes in condition of those most likely to utilise the system, and that the ability of older post-stroke patients to engage with the game is a positive indicator that motor control tasks such as those employed within the Asteroid Game are suitable mechanism by which interactive therapy can be introduced.

4.10.7 Discussion

In any physical rehabilitation routine, participants will be required to complete some specific task, and be instructed on how to perform this in some manner, whether vocally, or as written instructions on paper or computer screen. The trials performed using the software outlined in this chapter has shown that the specific details of these instructions is vital to the understanding and ability to successfully complete the desired actions. Trial participants in two groups were asked to perform the exact same actions, within the exact same software simulation, yet the performance between each group has been shown to be markedly different, depending on whether the task is presented to the participants as two separate actions (move to the target, follow the target), or a single sequential action (follow the target).

It may at first be considered counter-intuitive for the more 'granular' instructions to lead to a consistently worse performance, but this result is in-line with a growing body of literature related to action learning. Bernstein [Ber67] argued that action goals were mentally interpreted as a specific pattern to execute in 3D space, rather than a replication of a specific set of muscle patterns. This theory can be demonstrated in practice with one's own written signature, in that its basic unique pattern can be replicated whether the signature is written at a large scale utilising the shoulder and arm, or at a small scale utilising only the wrist and fingers.

Klapp and Jagacinski [KJ11] provide an overview of recent research into the concept of action goals being represented as an abstract code, rather than a specific list of sub movements to be completed by specific muscle actions. Their work describes learned combinations of movements as 'chunks', and posits that these chunks are processed into correct motions as and when required. Klapp used this terminology to describe motor learning for speech articulation [Kla03], in which two groups of experiment participants were encouraged to say pseudowords, having been instructed to either concentrate on separating out each syllable of the pseudoword or combine them into one single word. For the group instructed to treat each syllable separately, reaction time increased as a function of the number of syllables, which was not the case for the group encouraged to treat the word as a whole; this suggests that learning a word as a whole combines into one 'chunk' of motor learning, regardless of the number of syllables. This research therefore indicates that treating a combined action movement such as the one encoded within the Asteroid Game as a single combined action over multiple separate movements allows a more natural fit to the body's own motor control learning method. It is common in traditional video games for a 'learning curve' to be created by progressively adding new actions into the gameplay experience, so this finding has important implications for the concept of building motor rehabilitation tasks into such games.

The results generated using the framework are complementary to the existing corpus of research on motor task learning, adding knowledge on how 'chunk' learning affects manipulative tasks. Not only was transfer time prolonged in the group which had separate task instructions, but their accuracy at tracking the target was degraded, and was consistently poorer than that of the single combined action group. It is interesting to note that it was the latter of the two actions in the 'chunk' that was adversely affected, demonstrating that it is the process of creating the mental link of a 'chunk' that is impaired, rather than an individual motor control task. Works such as Piron et al [PTA$^+$10] indicate that video game enhanced motor control tasks can result in more robust improvements to patient ability, so determining which mechanisms have an effect, and how best to maximise their impact, is of great importance to gaming delivered therapies.

The results of the first trial bear similarity to the work of Orrell et al [OEM06], in which post-stroke participants performed better at a given balance task if allowed to implicitly learn how best to approach the task with mechanical assistance, rather that explicitly being shown how to solve the tasks. As their trial utilised post-stroke participants, that its results correlate to those discovered during the trial outlined in this chapter is evidence that the neurological mechanisms the Asteroid Game tested would elicit similar results in a trial of post-stroke participants. The preliminary results from the second trial also show that the rate of learning in the locking and tracking stages of target destruction is maintained in dominant hemisphere strokes; this provides some evidence to indicate that the game system is sensitive enough to detect changes in ability in those most likely to use rehabilitative software.

The trial results are significant, and provide a substantial proof as to the potential of rehabilitative games to assist as part of of a wider physical therapy, and the need for further research into how best to encode rehabilitation tasks within them. Patients can only spend so long engaging with even the most addictive and entertaining rehabilitation game, before detrimental effects start to build up. Firstly, the exertion of performing in-game physical movements for an extended period of time can result in tiredness, equal to that of traditional physical therapies [WKT$^+$13]. It has also been noted that those suffering from stroke have a higher occurrence of fatigue [IEP99], a mental or physical exhaustion that can affect the ability to complete physical therapy sessions [MJM05] even in younger patients [RLM$^+$03]. Boredom, too, can play a part in the efficacy of a therapy; a careful balance must be maintained between introducing challenges for the player to engage with, and not making those challenges too frustrating, so as to keep the player's interest [PRR10]. This will be of particular challenge in physical therapy games, where the limited number of

gameplay actions and experiences that can be generated with the limitation of being medically beneficial to the player may result in boredom over time, although the literature discussed in Chapter 2 does suggest that engagement is generally higher with video games than traditional therapies.

Due to factors such as boredom and fatigue, it is therefore important to ensure that the video game delivered rehabilitation sessions are utilised as effectively as possible so as to minimise any negative side effects. The research outlined here demonstrates that an incorrectly presented task not only reduces short-term ability at successfully replicating required actions, but can have long term effects, as the performance 'plateau' reached can be lower than that of a well presented task. In a theoretical rehabilitation game where in-game tasks aid in the replication of 'quality of life' movement actions, it is intuitive to see that a poorly designed task reduces the potential 'real world' outcome of a therapy, and could in fact be damaging to health. That the second trial indicates that the system is sensitive enough to determine if older post-stroke patients are undergoing the process of motor learning via playing a game is a sign that video game delivered therapies could be used to monitor patient progress as part of their rehabilitation routine, and that the game could provide an ongoing challenge to patients by reacting to such changes.

4.11 Conclusion

This chapter has detailed the creation and testing of a framework for aiding in the development of rehabilitative games. It allows for the creation of simple interactive real-time games with graphics and sound, and supports a number of different force sensing devices. The framework is almost entirely data driven, with a C++ core interacting with a number of data files that allow for the rapid creation and modification of in-game elements, such as the moving objects that make up the interactivity of the game, as well as the variables used by the rules of the game itself.

The framework works as intended, allowing for rapid development of simple games with which investigations into how best to provide rehabilitative benefit in a video game context can be undertaken. The modular and configurable nature of the framework has allowed for a variety of gameplay scenarios to be tested, and for changes to be made as necessary. This allowed for multiple scenarios to be tested, resulting in the 'Asteroid Game' - a game controlled by force-based input devices that presents to the player linked motor control tasks, requiring limb coordination to complete to a high level.

The game developed using this framework has enabled two significant results: that presentation of a sequentially linked task plays a significant role in mastery of that task over time, and that any improvements in ability to complete this task over time can be detected by the system in older, post-stroke patients whose performance is inhibited by their paretic limb. Together, these indicate that therapeutic video games derived from the game outlined in this chapter could form part of an ongoing therapy routine, allowing detection of changes in ability over time, which, if combined with correctly presented motor control tasks, could result in beneficial improvements in patient ability.

As of the time of writing, data recorded from trials is still being analysed within the Institute of Neuroscience, focussing on whether performance within the Asteroid Game over time is a positive indicator as to the success of a physical therapy, so there is potential for this work to enable further significant advances in the field of rehabilitative gaming, as such a link would allow for correctly designed games to be a measure of patient ability comparable to traditional standardised measures such as the CAHAI score or Wolf Motor Function test. The preliminary results outlined in Section 4.10.6 suggest that video games with sufficiently high quality input handling are sensitive enough to measure motor learning of sequentially linked actions, and as such could become a useful tool in the measurement of patient quality of life, as well as an entertaining method to engage with physical therapy.

Chapter 5

Detection And Success Determination Of Patient Movement In Rehabilitative Games

5.1 Introduction

The previous two chapters have covered methods by which the quality of a motion device can be determined, and how best for a patient with acute stroke to engage with a video game delivered rehabilitation routine while providing optimal health benefits. The purpose of this chapter is to build on these concepts, and cover methods by which therapists can create video game delivered therapies to patients suffering from a wide-range of debilitating conditions, utilising motion sensing devices to determine therapy engagement and success.

The investigation into academic works relating video games for rehabilitation performed in Chapter 2 shows that current research has largely focussed on either bespoke games created in an academic environment, or in the use of an existing commercial game as-is, with the specific outcomes of the research measured externally by the attending research and medical staff. There appears little engagement from the video game industry in creating games capable of being used in bespoke therapies, or that support measuring player input to a sufficient quality to allow a medical assessment. Without the support of the wider games industry, games created in a research context will likely suffer from the lack of expertise in many aspects of a successful game's development, such as game design, art and sound asset creation, and external quality assurance testing, all of which combine to form a more engaging, positive game experience. It would therefore be beneficial to promote collaboration between the games industry and medical experts, so as to create bespoke software of equal quality to commercial game releases, but also with medically valid benefits, affording their inclusion in physiotherapy routines.

The main focus of this chapter is on the creation and implementation a system capable of recording physical movements using motion devices, which can be combined to form a physical therapy program. The ability for someone to correctly replicate such movements is also part of the system, allowing for a measurement of ability over time. The software has been designed to be suitable for movement detection with a number of input devices, including those described in Chapter 3, allowing therapists to record 'gestures' for their patients to attempt in their own time using commercially available hardware. To facilitate and encourage the integration of such rehabilitative gesture detection to commercially available video games, the software presented within this chapter has been designed as a completely 'plug and play' solution; the software abstracts and encapsulates the testing of gestures and the communication with a variety of motion devices from the engine, allowing movement-based rehabilitation routines to be embedded within games without requiring context-specific knowledge. The software has been designed in a cross-platform manner, and has been used by a number of established local games studios, including Coatsink (www.coatsink.com) and Nosebleed Interactive (www.nosebleedinteractive.com), to develop games for the rehabilitation of stroke across both the PC and Android platforms.

5.2 Justification

Physical therapies designed to aid in the recovery of limb movement post-stroke are a combination of repetition of simple tasks [EC13], where a patient might be asked to raise and lower their hands a number of times, and of completion of gestures designed to replicate and gauge ability in 'real world' movements, such as ability to grip and lift a cup, or button a shirt. While a video game delivered therapy cannot be a complete replacement for the wide range of therapies and observations provided via the one-to-one care a trained therapist can provide, it is certainly possible that it can provide an entertaining method by which physical tasks can be presented to the player, and assessed for performance improvements that might suggest an improvement in limb function. While the work in Chapter 4 has shown the ability to detect and assess simple motor control tasks designed for acute hemiplegia, that leaves a wide range of motor control tasks that could be transferred to a video game environment.

To allow the detection of gross motions such as reaching and grabbing, and combined actions such as pouring a glass of water, a piece of software must be capable of tracking a player's ability to replicate such movements, and verify that they are a match to those intended. By performing analysis of a patient's ability to perform a

particular gesture, a system should be able to provide a more fine-grained result than simple pass/fail, as rehabilitation can be a long process, and any small improvement over time in ability to perform a gesture can be seen as indicators of the success of the rehabilitation process.

While there are broad classifications used to describe the level of disability of someone post-stoke, such as the NIH Stroke Scale [GS97] the exact combination of physical therapies used to aid in their rehabilitation will be tailored to their needs by physical therapists, and will change in response to a person's ability over time. A video-game based therapy system must allow for a similar amount of customisation and modification over time to be a truly useful tool, either to therapists or patients.

5.3 Gestures

It can be assumed that the user of physical rehabilitation software will be required to attempt a number of physical movements, or *gestures*, each of which may be a simple individual movement (raise arms in the air), or a more complicated compound action (raise arms, make pouring action with right hand). These gestures will either be established exercises designed for the rehabilitation of a specific ailment (such as tilting the hand using the wrist as in NHS guidelines for wrist therapies [Hos]) and provided to patients via leaflet or training, or as part of a tailored series of exercises created by a therapist to assist in the rehabilitation of a specific ailment, as part of a long-term physiotherapy [nhsa].

The gestures designed for aiding patient rehabilitation will generally consist of some combination of position and limb orientation, designed to test and extend a patient's current ability, so a gesture created for detection in a video game context should be able to track both position and orientation in 3D space. Depending on the exact movement encoded within the gesture, additional metrics such as smoothness of movement may also be desirable - for example, being able accurately replicate a 'pouring water' gesture within the system, and for it to be able to differentiate between a jerky and smooth attempt at its completion may be important, as a subject's ability to smoothly complete such a move could indicate an improvement in independence and quality of life.

For the purposes of the work described in this chapter, only upper body rehabilitation techniques were considered, and thus only hand-held motion devices, such as those benchmarked in Chapter 3 will be considered and discussed. By tracking the position and orientation of the hands in 3D space, it is possible to make assumptions on the pose of the entire upper body; it is assumed that a player will not attempt to

'cheat' the system by incorrectly holding the controller. This makes hand-held motion devices capable of assisting in the rehabilitation of the wrists, arms, shoulders, and potentially the spine; the ability to accurately and reliably detect finger movement is beyond the capability of most devices (for example, the newly released Kinect 2 can only detect whether a hand is in an open or closed pose), making the force-sensing devices discussed in Chapter 3 more suitable for such detection. The software has been shown to work with a hardware system that can detect finger movements, but this has some caveats that will be discussed later.

5.4 Representation of Patient Movement

There are existing methods by which to determine the similarity of two movement sequences, including the Fréchet distance, a measure of similarity between curves commonly used within handwriting detection algorithms [SKB07]. The correctness of a movement can also be tested via state machines [SBU+14], whereby a gesture attempt advances through the machine by testing against conditions such as controller separation distance, and orientation, and ultimately providing either a pass or fail result, as well as via machine learning algorithms [NS03][KSL07][HVL10].

Ideally, a rehabilitative video game should have a system whereby bespoke therapy routines can be added that are unique to a particular patient. Such gestures require input from medical staff such as occupational therapists, who specialise in determining the best rehabilitative regime for each patient. A bespoke regime can be built up for a patient using a combination of moves contained within a 'library' of prebuilt gestures, and of those created specifically to meet the unique needs of an individual. In order to facilitate the creation of new moves for a patient, a motion gesture system would ideally allow new moves to be recorded, and previous moves loaded and modified as necessary. Such requirements place limitations on the types of mechanisms used to record and test rehabilitative gestures. State machine systems are not particularly user-friendly, for example: even in such 'friendly' interfaces as the popular Unity game engine (www.unity3d.com; *Unity Technologies*), the creation of a state machine can be an obtuse combination of graphs and numbers, meaningless to those without appropriate training. Machine learning based systems tend to allow determination of which gesture from a precomputed pool is most likely being performed, without necessarily providing any information as to how well the gesture is actually being performed, something of key importance in physiotherapy, where the ability to perform a given gesture is an indicator of improved ability.

5.5 Waypoint Systems

An ideal system would allow the parameters a gesture is comprised of to be modified and visualised as necessary to create and customise bespoke rehabilitative routines, with an easy to use interface, and no prerequisite knowledge of underlying detection algorithms or computing science concepts. Treating gestures as an ordered collection of points in 3D space that a user must get the held controller close to provides an intuitive model, allowing for easy graphical representation of gesture parameters during gesture creation and editing. Positions that the user must reach can be easily defined as points within a 3D volume representing the space around a limb, while how closely a user must replicate these positions in order to be considered successful at the move can be represented as a shape around those points. Together, this position and shape, known as a *bounding volume*, along with any additional metrics that may influence the success of a gesture attempt such as orientation, required velocity and smoothness, are known as a *waypoint*.

Being a physical representation of the 'shape' of the move, waypoints are an intuitive method for the representation of a gesture. A graphical interface can intuitively show the movement as a 3D representation on screen, with positions as markers, bounding volumes as boxes or spheres, and the ordering of waypoints to replicate a gesture shown as arrows or lines connecting waypoints together.

Representing gestures as waypoints has the additional benefit of easy determination of important sections of a gesture. The difference between a gesture that requires the user to simply raise their arms high from any starting position, and one which requires the user to start in a very specific position can be easily ascertained when looking at the waypoints that comprise a gesture. Figure 5.1 demonstrates this: both gestures require the player to place their hands outwards, but while the left gesture has only one waypoint per limb, suggesting a loose requirement for how the gesture must be started, the right movements has a number of ordered waypoints, and specifically requires the user to start with their arms by their sides to pass.

5.5.1 Normalisation of Waypoint Systems

In order to correctly detect the success of a movement attempt by a patient, the movement must be matched to their unique physical attributes, such as shoulder width, and arm length. Figure 5.2 shows how even a simple gesture such as holding hands outwards can fail when motion control data is compared directly against prerecorded data - a gesture recorded by Player B could be impossible for Player A to complete.

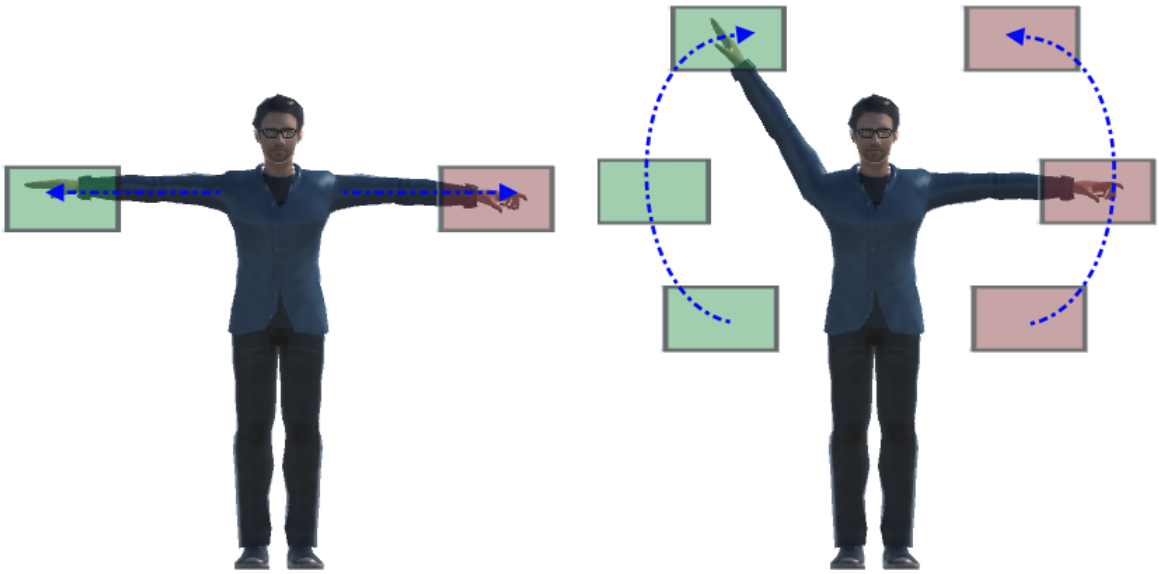


Figure 5.1: Left: Waypoints representing a simple 'arms out' gesture. Right: Waypoints that represent a more complex upward motion

In order for gesture recording and testing to be compatible amongst differing people, it is therefore necessary to adjust the waypoint positions and volumes to match the player attempting the movement. This can be achieved by normalising both the positions and volumes of the gesture, and the positional input received from the motion devices as the player moves. To facilitate this normalisation, the extents of a player's reach in all three axis must be determined. This is easiest to achieve by requesting a player to 'calibrate' the gesture detection software at the beginning of a usage session. By doing so, a bounding volume of reachability for each limb can be determined for a player, by which all further recorded motions can be described in relation to, such that the extents in each axis range from -1.0 to 1.0. By undergoing the calibration process when a gesture is recorded for future use, it becomes trivial to then 'fit' the wapoints described in the normalised space of the recorder, to the normalised space of the player attempting to replicate that move.

The process of gesture normalisation makes it possible for a gesture recorded once be suitable for a wide range of people, including those with limited movement in a limb. Someone with severe impairment in a limb may be only able to move their arm or hand a few centimetres, but by normalising waypoint positions and bounding areas to the area of movement that can be reached, there is greater chance of a gesture attempt being recognised, which can then be presented to the player as a reward in the form of positive feedback from the game being driven by the detection system, encouraging greater engagement with the rehabilitation process. For those successfully regaining movement in an affected limb, the normalisation mechanism ensures that the gesture will always fit within the area of reachability of the user, constantly providing some degree of challenge over time.

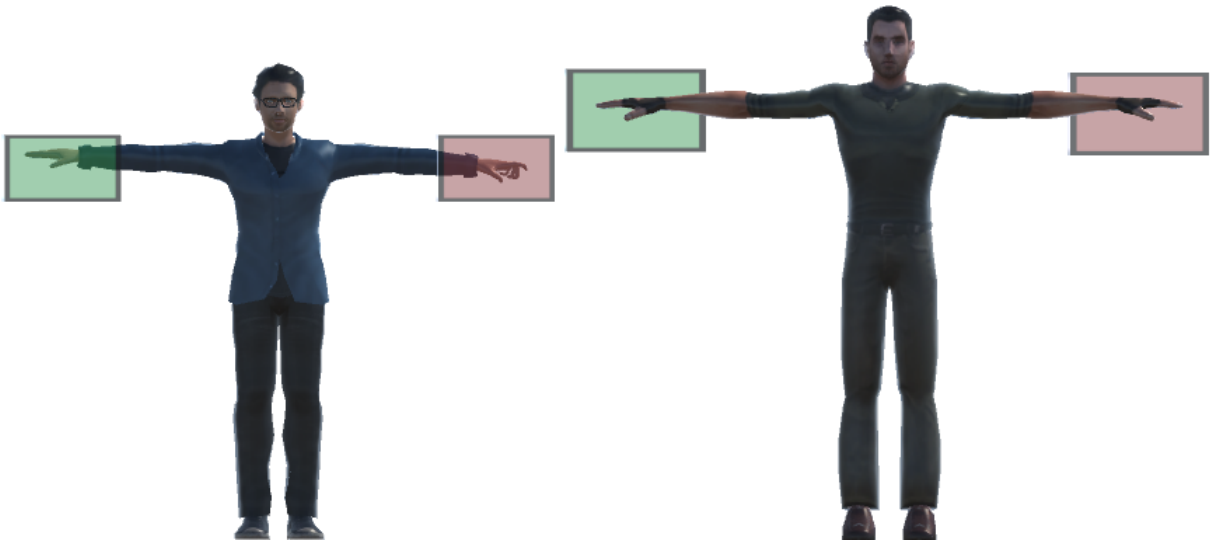


Figure 5.2: Normalisation of waypoints allows the same move to be used by multiple people. The arms of Player B (right) are longer than Player A (left), but normalising waypoints to their reachability allows the same recorded move to completed correctly by both players.

5.6 The Motion Control System

A full system for integrating patient gesture recognition into gaming software requires a number of components - there must be some method by which to create and inject new gestures into the system as necessary, there must be a way of bringing such gestures together as a cohesive routine to be undertaken by someone requiring physical rehabilitation, and there must be a way of testing the ability for someone to complete such gestures, indicating success at the physical abilities the gesture was created to test. This work describes a number of programs which together provide all of these components, allowing for video games to be used as a driver for rehabilitative patient care.

These programs, together known as the *Motion Control System*, allow a healthcare professional to create and modify rehabilitative gestures using a number of commercial-grade motion sensing devices, and deploy them to a game utilising the system, where they can be introduced into the gameplay mechanics as necessary. The system, outlined in Figure 5.3, abstracts away both the difficulties associated with generating accurate gestures, and the programming required to interface with a motion device, making it easier both for therapists to create rehabilitation strategies, and video game developers to integrate them into their products.

5.6.1 Motion Device Interaction

All of the devices currently supported by the Motion Control System will output at least their position and orientation, at a varying number of updates per second per device. Each device has its own particular position unit and axis basis; for example

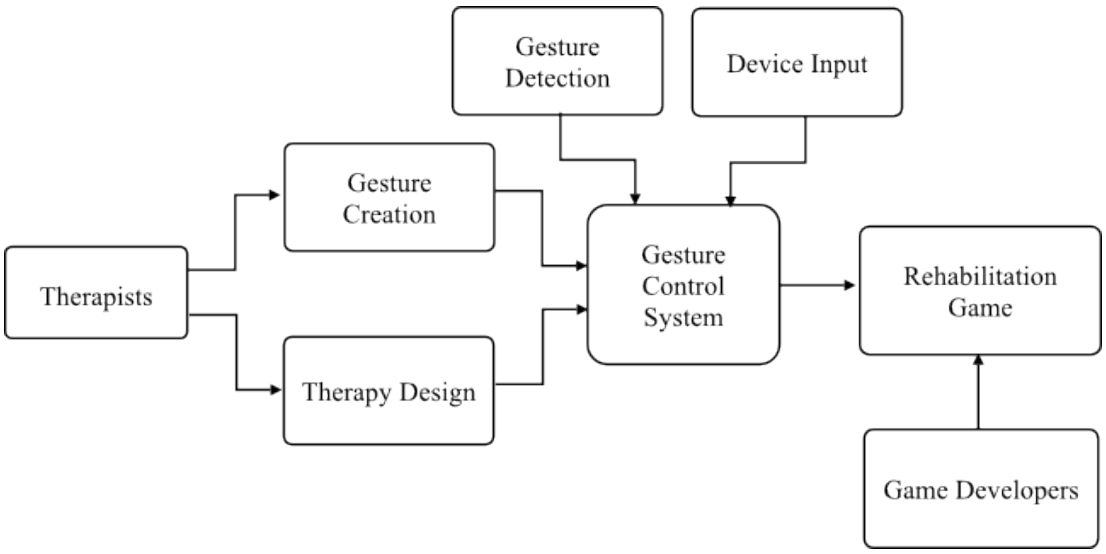


Figure 5.3: Overview of the Motion Control System: The structure of the system allows game developers and therapists to combine their skills to create engaging therepy-enabled games using motion controllers

the Sixense controllers have an output scale of 4 units per centimetre, while the Kinect devices output in millimetres, but has an orientation axis basis where the *Z* axis is 'up', whereas most devices use the *Y* axis. These differences require some degree of additional processing to rectify, so as to provide a 'common' data range for motion detection, that is agnostic of the actual device used by the patient. As part of the calibration process whenever a motion device is connected to the system, the user must stretch their arms outwards as far as possible in each axis, to form a bounding box that represents reachability. As well as 'normalising' waypoint positions, this also serves to normalise whichever units the device uses, such that -1.0 is always the furthest left, and 1.0 the furthest right the controller has been moved to. The basis axis for the motion devices are transformed as necessary into a 'right hand' coordinate system, whereby the *X* axis extends to the right, the *Y* axis extends upwards, and the *Z* axis points backwards; this matches the coordinate system used by the OpenGL (www.opengl.org; *Khronos Group*) graphics library, used for visualisation in the programs that comprise the system. By performing this coordinate transformation process where necessary, moves created by one device remain compatible with any other motion device capable of providing position and orientation of the patients hands, providing a universal system of gesture detection.

5.6.2 Representation Of Waypoints

Waypoints are represented as a 3-component vector for position, with a quaternion for the orientation, used due to their compact nature, and ease of comparison. The bounding volume of the waypoint is stored as a 3-component *extent*; that is, how far along each axis the bounding volume reaches around its position, measured in the same units as the position. This extent is assumed to be centred around the

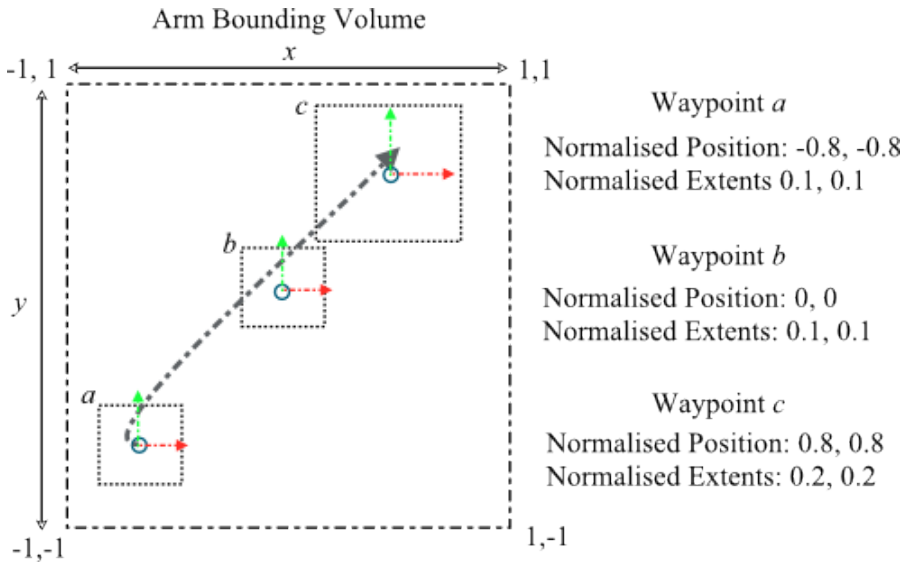


Figure 5.4: Forming waypoint volumes: The reachable area of each arm forms a bounding volume - waypoints can then be represented as a normalised position and size within that volume

stored position for the waypoint; together these form an axis-aligned bounding box, shown in Figure 5.4. These data structures are required for all waypoints recorded by the system. In addition to these, a number of additional details are stored in cases where the waypoint represents a more complex movement; an index to the special detection groups the waypoint belongs to is also stored; these additional data structures are discussed later.

5.6.3 Representation Of Gestures

Within the system, a gesture is considered to be a discrete action - detecting whether the user's arms are up in the air, performing a 'turn door knob' action, or lifting up and pouring a jug of water into a cup are all considered individual gestures, even if there might be sub-components to them. A gesture is stored within the system as an array of *streams*, with one data stream per motion device. Each of these streams will in turn have an array of waypoints, and a bounding volume within which waypoints are placed. A gesture may have a number of other structures which define a relationship between waypoints, such as those in separate streams that should be entered at the same time - such structures will be discussed further into the chapter.

5.6.4 Calibration Of Player Metrics

To facilitate the need to normalise player input as described in Section 5.5.1, a calibration stage occurs whenever the Motion Control System connects to a motion controller. The player is asked to perform the following actions, indicated via a series of pop-up boxes:

1. Place arms down by side

2. Move arms outwards into a T-shape

3. Move arms up in the air

4. Move arms out forwards

Performing these actions will cause a bounding volume for each motion controller to expand to match the volume of reachability, demonstrated in Figure 5.5. By having a bounding volume per controller, the system can then intuitively support gesture detection for people with impaired movement. Waypoints positions and bounding volumes will be automatically scaled to match the area in which the user can reach, as shown in Figure 5.6, allowing attempts to be made with poor, or asymmetric reachability with the limb. The time taken to calibrate, and the mean velocity of each sensor, are stored upon completion of calibration, as they are used during score determination as a factor by which to scale recorded gesture metrics, allowing someone with slow limb movement to still attempt gestures with an aspect of timing or speed.

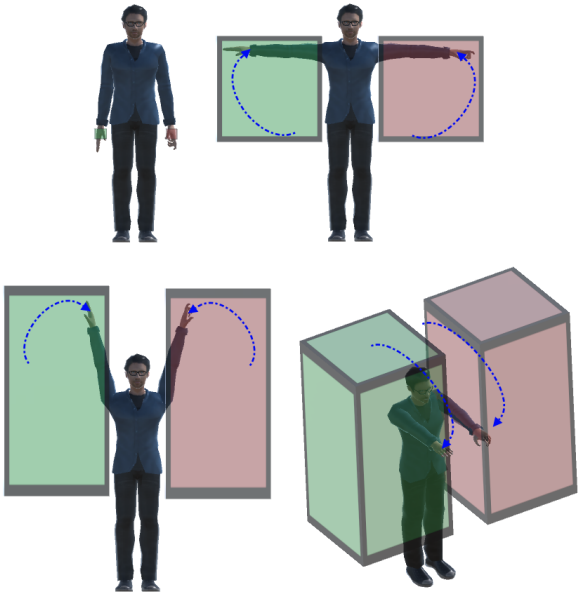


Figure 5.5: Bounding volume generation: By requesting that a player should move their limbs in all 3 axis, a bounding volume of reachability for each limb can be formed

5.7 Gesture Success Determination

Once a gesture has been recorded by the system and transformed into waypoint sequences, it can then be used to measure the success of attempts to replicate that movement. A gesture attempt is the process of the user moving their limbs in an

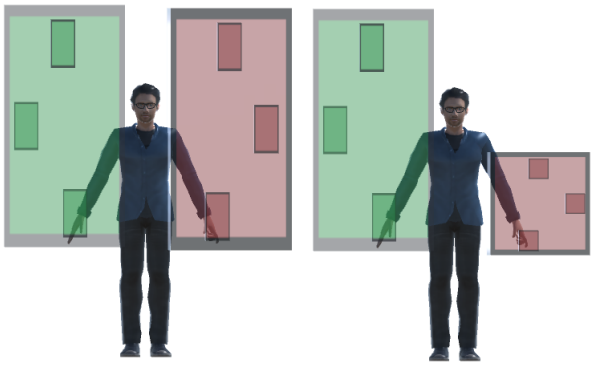


Figure 5.6: Left: Able-bodied player with their bounding volume, and gesture waypoints. Right: The same gesture is automatically reconfigured for someone with asymmetric movement ability

attempt to replicate the desired series of motions, which have been previously described to the user in some manner - an on-screen video perhaps, or through one-on-one tuition with a therapist. To correctly replicate the gesture, the user must perform the same actions, in the same order, as in the original recording - if a therapist's gesture is to raise the arms straight up, and then straight forward, then a 'successful' replication attempt must have both movements in the correct order. While a user may demonstrate increased ability or dexterity in performing a move incorrectly, it is assumed that the gesture has been designed for a specific reason, and so only accurate replications of the recorded move should be considered a success. As the rehabilitative gestures are assumed to test or push the user in some way, gesture success or failure should not be a binary 'pass / fail' , but a quantity that defines how well the gesture been replicated, so as to allow small improvements in ability to be reflected in patient feedback.

It is likely that a therapist will wish to see whether their patient can perform 'real life' everyday tasks such as using cutlery, opening doors, and pouring themselves a drink; many such actions are also included in standardised measures of ability that such as the Wolf Motor Function Test, indicating the importance of the system being able to replicate and detect them, as replicating such measures provides the means of measuring ability using well-known metrics. Such gestures may require additional context beyond attempting to be in a specified pose. For instance, jerky movement may indicate a likelihood of dropping items, while some gestures may require coordination (such as picking something up with one hand and passing it to the other), meaning that not only is being able to move each limb is important, but how they move in relation to each other is, too. Therefore, it is necessary to take into consideration factors such as the smoothness of the motion, and how each limb's movement matches the overall purpose of the gesture, when calculating how well a movement has been performed.

Determining the success of a gesture replication attempt can be broken down into analysis of a number of different metrics relating to the user's limb movements throughout the gesture attempt, each of which must be recognised and measured by the system:

1. **Shape**: Has the overall 3D shape of the gesture been matched?
2. **Orientation**: Was the limb in the correct orientation throughout the attempt?
3. **Smoothness**: How smooth was the movement throughout the attempt?
4. **Speed**: How fast was the movement completed?
5. **Synchrony**: Were the limbs moved in synchrony throughout the attempt?

Depending on the context of the gesture, not all metrics may be relevant, so their measurement should be enabled on a per-gesture basis. Between them, the metrics fully define the spatiotemporal relationship between the limbs and the body, such that following the therapist's intended actions exactly can only result in success in each enabled category.

5.7.1 Waypoint Sequences

Once recorded, a gesture can be thought of as a number of ordered lists of waypoints - each controller has its own list, and each list may contain a different number of waypoints, defined by however many waypoints are required to fully capture the intent of that limb's role in the gesture. The ordering of the waypoints within the list is important, as it defines the overall shape of the gesture. Take for instance the simple gesture shown in Figure 5.7, in which each limb has 3 waypoints in its waypoint sequence; it is only the specific ordering which defines whether the gesture requires the user to have their arm up and move it downwards, or start with their arm down and move it upwards. The system stores gestures as a an array of waypoint structures for each controller recorded, that are assumed to be in sequence, such that the first element of the array is the start of the sequence. There is no requirement that each array be of the same size - for gestures that only test a single limb, such as turning a door knob, it may even be the case that a controller has no waypoints at all - in which case the user is assumed to be allowed to move the limb freely.

5.7.2 Detecting Waypoint Shape and Orientation

Of all of the success metrics outlined in the previous section, shape and orientation are the most intuitive to visualise. As the user attempts a gesture, their controllers

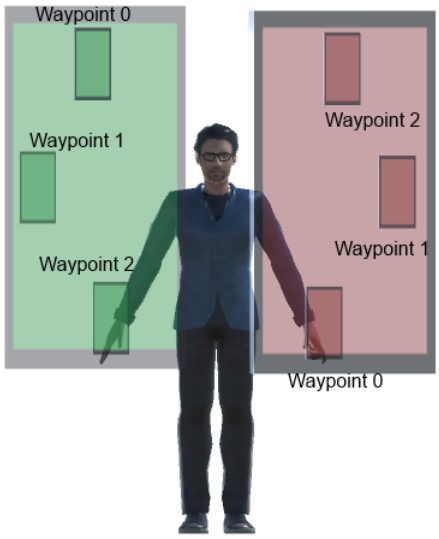


Figure 5.7: Gestures are defined by ordered lists of waypoints. In this example, the ordering creates either a 'move arm up or 'move arm down' gesture from the same waypoints, indicated by the numbered waypoints for each limb

must pass in turn through each waypoints in their list, with the bounding volume of the waypoints determining whether the controller is close enough to be considered to have successfully replicated the gesture - the bounding volumes essentially create an intuitive positional tolerance at critical sections of the gesture. The controllers should additionally be in the correct orientation: whether the palms are level to the ground, or sideways, is an important decider in whether a 'handshake' gesture has been completed successfully, for example.

As most rehabilitative gestures are by their very nature repetitive, there is some leeway afforded in determining the success of a particular attempt at a gesture; if a user feels they have incorrectly performed a move, they are likely to start again without explicit prompting, allowing the system to pick up the start of a gesture again to determine its success. For this reason, the system only checks waypoints further along the list when tracking a gesture, meaning that any skipped waypoints are considered failures, reducing the potential shape score of that gesture attempt. Figure 5.8 shows cases where a user can miss a waypoint, and demonstrates why such waypoints should not be further considered in a gesture attempt: here, the player has missed waypoint two, but can only re-enter the waypoint by performing a movement that does not match the intent of the waypoints. They should therefore not be allowed to improve their score by 'filling in' waypoints by moving incorrectly. A side effect of this waypoint limitation is that the detection requires less processing, which is of benefit when considering that the system could be deployed on low-end hardware, such as Android-powered gaming devices like the OUYA console (www.ouya.tv; *Razer Group*).

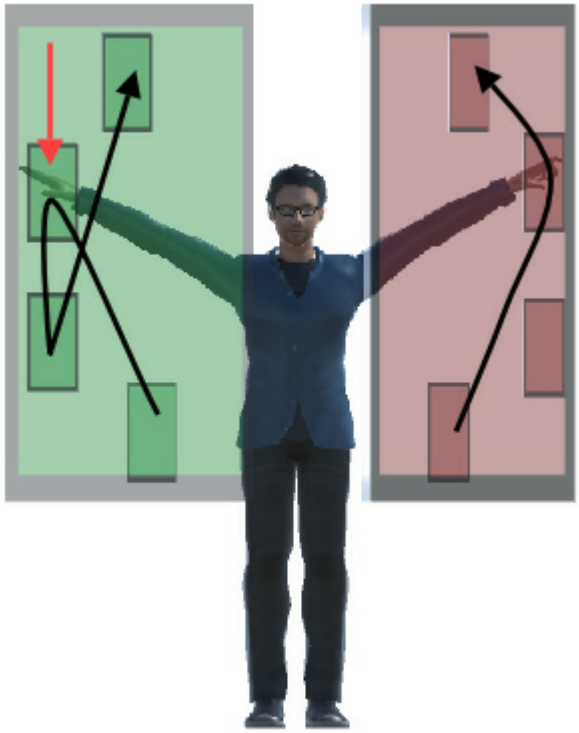


Figure 5.8: Fail cases in a waypoint sequence: The user's arm movement (indicated by the black arrow) has not passed through the waypoints in the correct order; the user should not be able to 'fill in' the missing waypoints by continuing to perform the movement incorrectly

Along with the orientation of the controllers as they enter each waypoint in the sequence, it is also necessary to take into consideration the direction of travel of the controller. As it is theoretically possible for a motion device to move outside of the waypoint volumes without penalty (barring the penalising areas described in section 5.7.3, it is therefore also possible for the next logical waypoint in a sequence to be entered from a direction other than that made during the initial recording. Figure 5.9 (left) shows an example of this, with the solid line representing the intended path through the sequence of wayoints, and the dashed line the path taken by the user during their gesture attempt. For the purposes of success determination, a waypoint entered from the wrong direction is considered a failed waypoint, which impacts on the total shape metric, and increments the detector to the next waypoint so as to discount the waypoint from the gesture attempt.

The correct direction can be inferred from the normalised direction vector d^0 between a newly entered waypoint and the previous. By comparing d^0 against the direction vector d^1 between the centre of the newly entered waypoint and the controller position, a simple dot product test can be performed, whereby the waypoint is allowed if the dot product is greater that 0.0. Example pass and fail cases are shown in Figure 5.9 (right): a controller entering waypoint 3 along direction a is considered valid, while direction b is not.

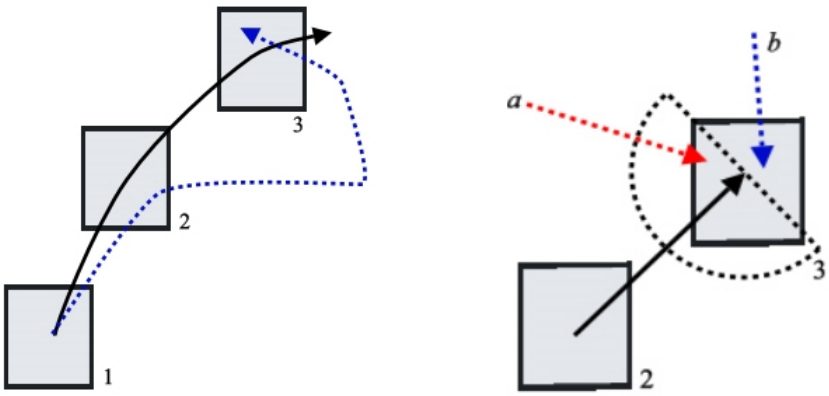


Figure 5.9: Left: An example of an incorrectly passed waypoint sequence, with the user's path (dotted arrows) missing waypoint 2. Right: By comparing the direction vector between waypoints, and the direction the controller entered a waypoint, it can be determined whether the user's movement has matched the intended movement. In this case, the dot product between the waypoint direction and waypoint entry vector *a* is greater than 0.0 and thus passes, while entry vector *b* will not, due to a resulting dot product of less than 0.0

5.7.3 Deadzones

It is possible that certain gestures are designed to test success at a very specific gesture, with little room for leeway. In such circumstances, it may be seen as undesirable to allow the motion controllers to enter certain areas of a player's overall bounding volume for the duration of a gesture attempt, beyond the impacting of shape score. For this reason, additional waypoints can be added to a gesture that are marked as 'deadzones'. Such waypoints will cause a movement attempt to fail instantly if a controller enters them, but otherwise do not contribute towards the final score for shape upon score calculation.

5.7.4 Waypoint Groups for Speed, Synchrony, and Smoothness

The waypoints that make up a gesture are grouped together within a number of constructs that aid in the correct determination of the correctness of the gesture attempt by the system. Most obviously, waypoints representing the movement of a particular motion device are gathered together. Waypoints are further split up into streams, with one stream of data representing the sequential ordering of the waypoints of one device.

Each gesture can also have a number of *synchronisation*, and *speed* groups, containing 0 or more waypoints from any of the recorded devices. These additional groupings are used to provide success measurements of the parts of a gesture that represent complex actions requiring synchrony between limbs, or smoothness of motion. Whether a gesture requires any of these optional additional elements is up to the therapists recording the the motion; a gesture designed to test whether a pa-

tient can move their arms up above shoulder height at all probably doesn't require any consideration of the time taken or the level of synchrony between limbs in order to accurately provide a success metric, while for a motion such as simulating the ability to pick something up with both hands, synchronisation of limbs is vital to whether a gesture attempt has been successful or not.

5.7.4.1 Detection of Movement Synchrony

Synchronisation of movement between limbs is often an important part of a rehabilitative gesture. Take for instance a case where the gesture is designed to measure the ability for someone to pick up a cup with one hand, and pour water from a jug into it; successful completion of such a movement could be used to determine the ability of someone to complete everyday tasks unaided. This example movement obviously contains a high degree of synchronisation and coordination between the limbs, as performing the actions out of sync would result in spilled liquid were it performed in reality, a potential hazard and indicator of an inability to function independently. Due to the importance of accurate limb coordination, it is often part of the standardised tests used to determine rehabilitation progress; for instance, the Wolf Motor Function Test has a task specifically for testing the symmetry of movement in both arms, by asking the patient to fold a towel (page 25; [TMC$^+$], while the CAHAI assessment includes tasks utilising both arms in coordination, such as wringing out a wash cloth (page 26; [BSM$^+$]).

While it is necessary to determine that limb coordination is taking place for certain movements (both to replicate 'real life' movements, and those specifically used in standardised tests), it is possible that not all sections of a gesture attempt require that the movement mimics the original recording's positions and orientations precisely down to the second. Take for instance the water pouring gesture outlined above; by splitting it into three subsections as follows:

1. Pick up and bring cup and jug to chest level
2. Pour water from jug into cup
3. Put cup and jug back down

It can be seen that not all of the whole gesture requires limb coordination: the first and last phases are independent movements, with only the middle placing an importance on synchronisation of movement between limbs. It would therefore be inappropriate to have a system whereby the user is penalised for not performing these independent sub-movements exactly as recorded, and so there must be a mechanism within the system by which only certain parts of a gesture are considered as explicitly requiring synchrony.

To allow such synchronous gesture areas to be created, each gesture can have a number of *Synchronisation Groups* - a named list of waypoints from any of the controllers participating in the gesture. When a gesture is attempted, all of the synchronisation groups within it are considered to be in an 'inactive' state. A particular synchronisation group can be configured to be 'triggered' into an active state when a motion device is moved into the bounding area of one of the waypoints assigned to that group (so-called 'On Enter' groups), or to be triggered when a controller moves from a waypoint in the group, to one which is not in the group ('On Exit' groups). When a synchronisation group is triggered, an internal timer is began, within which all motion devices with waypoints present in the synchronisation group must either enter or exit the group. Each group can then be configured to either cause a score penalty, or to outright cause a gesture attempt to be considered a failure if too long is taken.

The synchronisation group mechanism allows a varying level of granularity in the coordination required to complete the move - a whole subset of the move could be grouped together to ensure that all controllers begin and end that subset at approximately the same time, or individual waypoints could be paired to enforce a more strict synchronisation. The ability to give names to these groups can be used to help convey what the group is testing for, or differentiate it from other synchronisation groups when modifying the gesture.

5.7.4.2 Detection of Movement Speed & Smoothness

Much as with synchronisation, there may be a requirement for a gesture to have restrictions on speed and smoothness of movement. The therapist's subjective opinion on a patient's speed and fluidity of movement are used as part of the scoring metrics in measurements such as the Wolf Motor Function Test [TMC$^+$], so the ability of the system to provide a similar metric during gesture attempts would be of benefit. If we refer back to the previous example of water pouring, it can easily be imagined that we may want the second section to have a requirement for the movements to be handled smoothly, as excessive movements could result in water spillage if actually performed. There may also be cause for speed to be a consideration: a therapist may decide that measuring someone's ability to raise their hands within 5 seconds will help them make judgements on a patient's progress over time, or a simple to understand goal for a patient to work towards.

To facilitate gestures with these requirements, the concept of a *Speed Group* is introduced. As with Synchronisation groups, these are named lists containing waypoints from any of the streams that make up the gesture, and will only become active once

a device has entered the bounding volume of one of the waypoints in the group. Active groups can measure one or both of the following: the mean velocity of the devices that have waypoints within the group, or the *jerkiness* of the devices. In this context, jerk is the third derivative of position: it measures the change in acceleration. A smooth, completely linear motion will display zero jerk, while an erratic motion that stops and starts will have a larger amount of jerk.

Both the smoothness and speed metrics utilise the stored results of the initial calibration data - the time taken to complete the calibration, and and the mean amount of jerk present in the controllers while doing so, is used as a baseline measurement. Jerk is calculated by storing all positions during the calibration stage, normalising them post-calibration, and then determining the mean of the third derivative. This results in a 'normalised jerk' value, which can be compared against between the gesture recording and following gameplay sessions.

With these values, the success at passing through a Speed group can be ascertained - a normalised jerk score outside of a stored range will cause the group to fail, as will taking a length of time outside a stored time range to complete the group, scaled against the time taken to complete calibration.

5.7.5 Determining Gesture Attempts

The repetitive nature of the gestures the system is designed to detect affords some leniency when detecting the relative success of a gesture attempt. The detection of a gesture attempt is comprised of three stages:

1. Start Phase (wait for a device to enter a starting waypoint)
2. Determine waypoints hit during movement, and their group effects
3. End Phase (Score calculation, pass / fail)

A gesture attempt is deemed to have begun when the first waypoint in the sequence of one of the motion devices is hit, and deemed completed when the last waypoint of every stream has been hit; movements before or after these points bear no impact on scoring, allowing players time to prepare or recover from gesture attempts. A gesture attempt will automatically enter the end phase if no new waypoint has been entered for a preset amount of time, to accommodate instances where the player stops playing due to fatigue. While the first and last waypoints must be entered to trigger the start and end phases of a gesture attempt, waypoints within the sequence may be 'skipped' by not following the gesture closely enough, impacting on the final score of the gesture attempt. As the first and last waypoints are

triggers to begin and end a gesture attempt, they should be made as easy to successfully complete as possible - when testing the system it was common to increase the bounding volume extents of the start and end of a movement, as for the most part it is the movement throughout a gesture that is the indicator of replication success, not the start and end points.

5.7.6 Scoring Gesture Attempts

When a gesture attempt reaches the end phase, either by hitting the last waypoint for each device, or by timing out, a score for the attempt must be generated by the system, and whether the attempt has passed or failed determined. The purpose of the scoring system is not to punish the player by receiving a low score for poor attempts, but to provide a means by which to grade the relative improvement in performance over multiple attempts and play sessions.

5.7.6.1 Shape

The primary scoring metric for a gesture attempt is the overall shape of the movement compared to the recorded gesture. The scoring for shape is derived from how many waypoints in a device sequence are hit; by the following simple formula:

$$Shape = \frac{\sum_{i=1}^{s} \sum_{j=1}^{w} f(i,j)}{\sum_{i=1}^{s} w_i} \tag{5.1}$$

Where s is the number of waypoint streams, w is the number of waypoints for a given stream, and f being the following scoring function, where $dist$ is the shortest distance between the motion device and the waypoint centre:

$$f(i,j) = \begin{cases} 0, & \text{if waypoint missed} \\ 1, & \text{if distance} < 0.33 \\ \frac{((dist-0.33)/(1-0.33))}{2} + 0.5, & \text{otherwise} \end{cases} \tag{5.2}$$

The scoring function uses the closest position to the centre of a waypoint bounding volume a controller reached before exiting the waypoint to determine its result: the position is normalised against the bounding volume extents, and will return 1.0 if the distance is less than 0.33, and will linearly degrade to a value of 0.5 at a distance of 1.0. This allows a gesture attempt that only just enters each bounding box to still score well, while providing a means to differentiate an increase in performance.

5.7.6.2 Orientation

The orientation score is derived from the closest a controller got to matching the orientation of the waypoint while within its bounding radius. It uses the following equation:

$$Orientation = \frac{\sum_{i=1}^{s} \sum_{j=1}^{w} 1.0 - \frac{\theta}{90}}{\sum_{i=1}^{s} w_i} \tag{5.3}$$

Where s is the number of waypoint streams, w is the number of waypoints for a given stream, and θ is defined as the result of the following:

$$\theta = 2(acos(\|a \cdot b\|)) \tag{5.4}$$

Where a and b are the waypoint orientation and controller orientation as normalised quaternions, $\cdot$ is the dot product operation, and *acos* is the arc cosine in degrees. By clamping the resulting angle θ between 0.0 and 90.0, a linear score between 0.0 and 1.0 for orientation will be derived.

5.7.6.3 Smoothness

Certain waypoints may by linked by a speed group that measures how smoothly the player moves through them. This is done by storing device velocities whilst the group is active (that is between the point at which the first device enters a tagged waypoint, and the point at which the last device leaves the last tagged waypoint, or the gesture fails), and from the resulting list calculating the mean *jerk* of the controller throughout the group.

$$Smoothness = \frac{\sum_{i=1}^{sg} (\frac{\sum_{i=1}^{j} jerk(i,j)}{j})}{sg} \tag{5.5}$$

where sg is the number of smoothness groups, j is the number of samples taken while the group was active, and *jerk* is the jerk recorded in that sample, minus the mean jerk stored in the calibration process. This will result in a score of 0 for failing all smoothness groups, otherwise a linear scoring towards 1 for zero jerk in all smoothness groups.

5.7.6.4 Speed

As with the smoothness score calculation, the speed score is dependent on the number of speed groups. The speed score is calculated as follows:

$$Speed = \frac{\sum_{i=1}^{sg}(f(sg))}{sg} \tag{5.6}$$

Where *sg* is the number of speed groups active in the gesture. If the speed group has failed (by not ever being triggered, or being completed too fast or too slowly), then that group will score 0.0, otherwise the score will be based on how long the group took to complete, compared to the minimum and maximum completion times specified for the group. The calculation for f is as follows:

$$time = \frac{(\text{time taken}) - \text{min time}}{\text{max time - min time}}$$

$$ideal\ time = \text{min time} + \frac{\text{max time - min time}}{2} \tag{5.7}$$

$$f = \begin{cases} \frac{time}{\text{ideal Time}}, & \text{if ideal time >= time} \\ \frac{\text{ideal Time}}{time}, & \text{otherwise} \end{cases}$$

Where *time taken* is the time taken to complete the group divided by the time taken to calibrate (in seconds), and *min* and *max time* being the stored time bounds of the move - rather than in seconds, these are factors of the time taken to calibrate when the move was recorded, allowing people with slow limb movement a chance to participate in a pre-recorded gesture, and creating a challenge over time, as the move will become more difficult if their speed increases. The calculation results in a player scoring 1 for taking a length of time equal to that half way between the minimum and maximum times, linearly degrading to 0 for being as fast as the minimum time, or as slow as the maximum time.

5.7.6.5 Synchrony

For synchrony groups, scoring is determined by the time (scaled by the calibration time, as with the speed calculation) taken for all devices represented within the group to enter, exit, or both, depending on the properties of the group. As with other components, the actual score is derived from the follow equations:

$$\text{enter time} = \frac{\text{(enter time limit - time taken to enter)}}{\text{enter time limit}} \tag{5.8}$$

And likewise for exiting:

$$\text{exit time} = \frac{\text{(exit time limit - time taken to exit)}}{\text{exit time limit}} \tag{5.9}$$

The final synchrony score is then calculated as:

$$Synchrony = \frac{\sum_{i=1}^{sg} \left(\frac{\text{enter time(i)} + \text{exit time(i)}}{x} \right)}{sg} \tag{5.10}$$

Where sg is the number of synchrony groups, and x is 1 in cases where only one timing trigger (either enter or exit) is enabled, or 2 in cases where both triggers are enabled.

5.7.6.6 Final Score Calculation

Once all of the separate metrics have been calculated, a final score can be derived from them, using the following formula:

$$Score = \frac{Shape + Orientation + Smoothness + Speed + Synchrony}{\alpha} \tag{5.11}$$

Where α is the number of metrics enabled for the move - for a move only graded by the shape metric, α would be 1.0, and for a move with all metrics enabled it would be 5.0. This metric takes into account whether or not each of the metrics was required by the gesture or not, but otherwise gives equal weighting to each of the used metrics. The final score will be between real value between 0.0 and 1.0 inclusive. The score allows separate gesture replication attempts to be graded relative to each other, but is not meant to be used alone as a medical diagnosis.

5.8 Motion Control System Development

The Motion Control System software described in this chapter has been developed to allow for both the recording of gestures by trained therapists, and the detection and grading of attempts at replicating prerecorded gestures. It must also be capable of being integrated into the existing code bases of video games, so as to enable them to be used as entertaining rehabilitation tools. To facilitate these goals, the system as been developed in a modular way, with ease of integration into other projects a core concept of its architecture. All of the software for the system was developed using the C++ programming language, due to its performance characteristics, the ubiquity of high-quality compilers across multiple platforms to generate optimised assembly, and the wide range of supporting libraries and toolkits compatible with the language. Figure 5.13 shows the basic structure of the individual modules that make up the system.

The core of the system is the *MotionLibrary* project, designed as a static library to link into other projects. It contains all of the code functionality necessary to load, save, and parse XML files containing routines and moves, interface with a

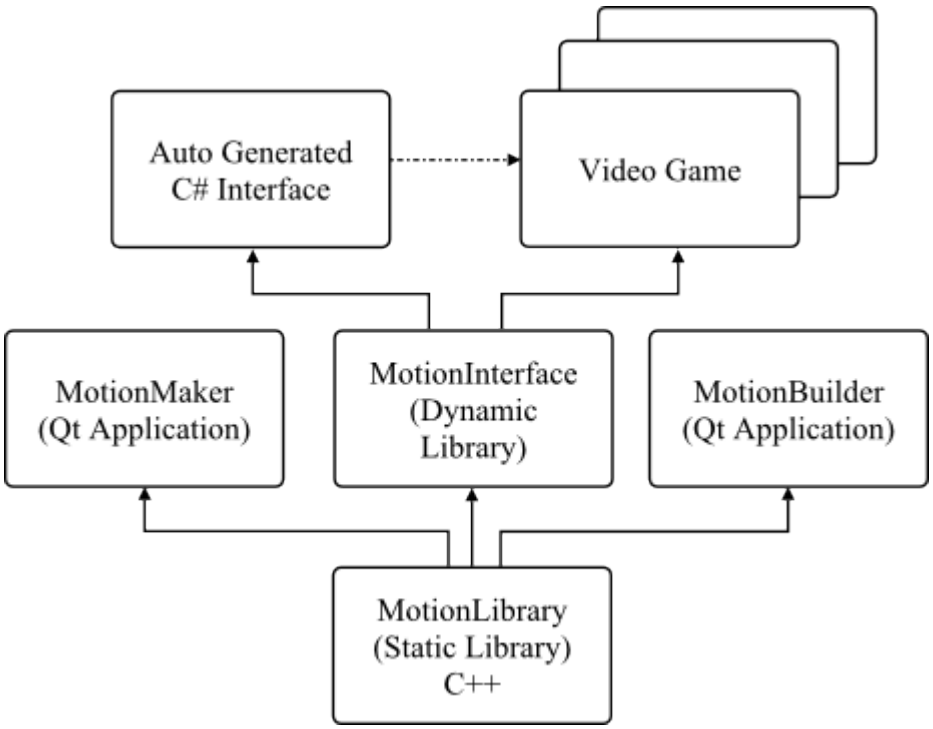


Figure 5.10: Decomposition of gesture system into a series of apps and libraries

range of motion devices, and detect gesture success using their output. The popular TinyXML library (www.grinninglizard.com/tinyxml; *Lee Thomason*) is used to parse XML files where necessary. This system library is then used by three further components: The *MotionMaker*, which handles creation of gestures, the *MotionBuilder*, used to package moves into a number of routines, and the *MotionInterface*, designed to be linked to programs designed for rehabilitative purposes. While the target platform throughout development was Windows, the library has been designed to be cross platform - both the *MotionMaker* and *MotionBuilder* applications use the popular *Qt* user interface toolkit, allowing for the programs to be compiled and deployed on additional platforms with relative ease.

5.9 The MotionInterface Library

The *MotionInterface* project is designed to allow the MotionLibrary to be accessed and utilised by other programs, such as game engines. As there is no common application binary interface for C++, access to the functionality of the library must be marshalled through a language that does - this is a necessity for a dynamic library to be accessed and utilised correctly by external programs, which may or may not be compiled using the same compilation rules as the MotionLibrary. To allow this, the MotionInterface project compiles a frontend to the system in pure ANSI C, allowing for correct extraction of function entry points by programs wishing to use the system, while still allowing for an intuitive object-oriented programming design internally.

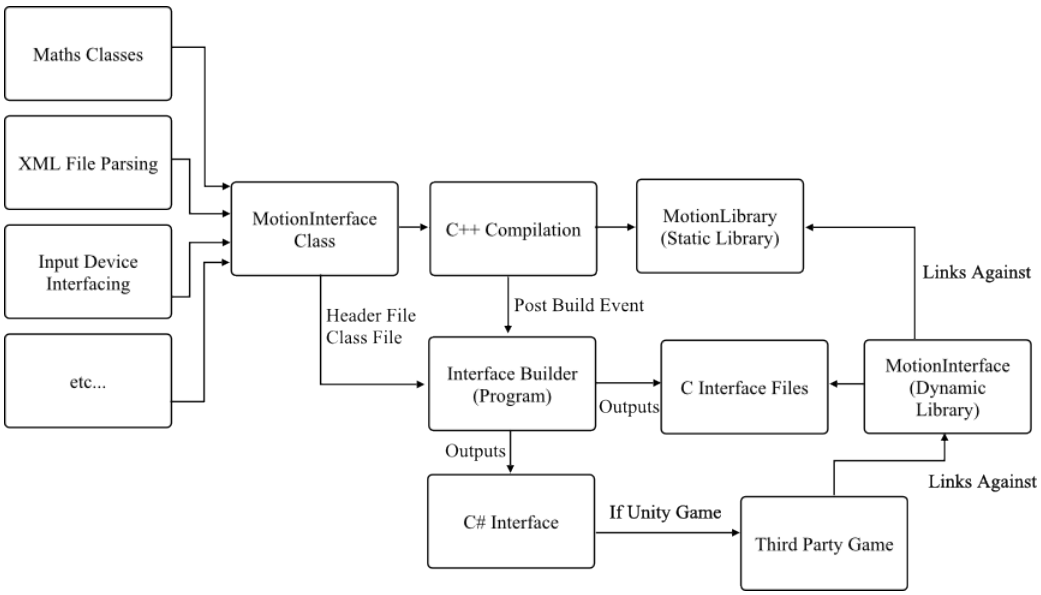


Figure 5.11: Generation of the MotionInterface library: C-style linkage is automatically generated from C++ classes via the *Interface Builder* program, which can additionally generate a C# import class suitable for use in the *Unity* game engine

This conversion from C++ to C is a two part process, outlined in Figure 5.11. A C++ 'Motion System' class is created, designed to work as a public API for the classes within the MotionLibrary - it provides functions for initialising the system and motion devices, loading gestures from files, beginning and ending gesture detection routines, and the methods by which to query to the state of devices, gestures, and detection attempts. While no client-side interaction is required to test for gesture success, access to such metrics allows for additional debugging if necessary, and visualisation of the gesture waypoints; during testing of the system it was found that a graphical representation of the movement being attempted on screen aided successful completion, even in cases where the player knew how to complete a stated gesture. Upon compilation of this API class, an external program is executed (when compiled using Microsoft Visual Studio 2013), which parses the header file of the class, and outputs three new files:

1. A raw C header with exported function declarations
2. A file of raw C functions calling the underlying Motion System
3. A C# class that imports functions from a dynamic library

The C files allow the MotionInterface project to be compiled as an easily linked dynamic library, while still allowing for modern object-oriented design principles to be used 'under the hood'. The C# class allows ease of integration of the library into game engines that support the popular language, such as the Unity game engine. Further discussion of the library and Unity is found in Section 5.13.2.

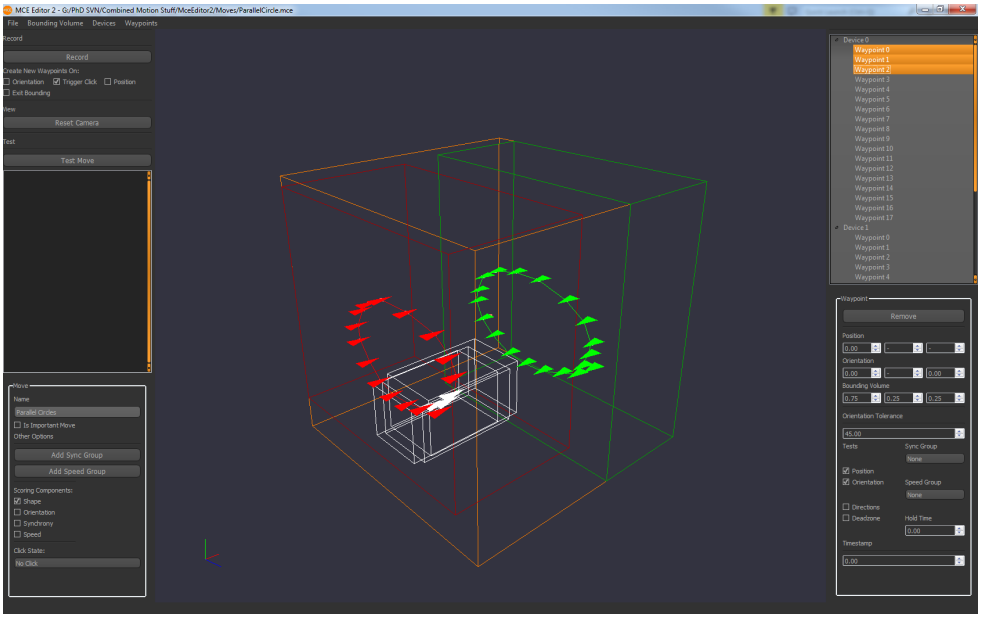


Figure 5.12: The *MoveMaker* application showing the creation of a 'rowing' gesture - arrows represent the centre of waypoints, with the direction indicating the ideal controller orientation

5.10 MotionMaker: Gesture Creation and Editing

To facilitate the creation and modification of gestures for use in rehabilitative video games, the MotionMaker program has been developed. It combines the classes within the MotionLibrary project with a Qt-powered user interface, and allows the saving, loading, and creation of gestures, as well as their visualisation and modification in real-time. The standard layout of the application is shown in Figure 5.12. The application follows the standard convention of having an upper toolbar from which to execute actions, with tabs for 'File' operations, motion controller connectivity, and help. The large central area of the application is used to visualise gestures in 3D space, while the panels to the sides of the screen are used to select waypoints and groups, and modify their properties.

5.10.1 Gesture Visualisation

Gestures that have been loaded or newly recorded can be visualised in 3D space, via an OpenGL enabled viewport. This view allows for the current gesture to be zoomed in and out of using the mouse wheel, as well as panned around via dragging the mouse with the right button depressed. All of the graphical information is placed with a wireframe cube on screen, which represents the normalised extents of reachability. The normalised bounding volume of each controller used to record the gesture is also displayed using a colour-coded cube. Within each of the device bounding volumes, the waypoints are displayed, as a coloured square to represent its origin, an arrow to represent orientation, and normalised bounding volume cube.

The relative ordering of waypoints is displayed using a line that passes through their origins. If there are motion devices currently connected to the system, their current normalised position and orientation will also be displayed.

5.10.2 Creating New Gestures

Gestures can be recorded in real-time, using any of the supported motion devices. When recording a gesture by pressing the *Record* button on the user interface, new waypoints can be added to the gesture in several different ways, selectable from a menu:

- Upon Trigger Click
- When a position delta threshold is reached
- When an orientation delta threshold is reached
- Automatically generated periodically

The normalised position and orientation thresholds, as well as the periodicity of automatic waypoints can be selected from a menu. The process of recording the position and orientation of the controller will continue until the *Stop* button on the user interface has been depressed. Waypoints added to a gesture at this stage will be created with a default bounding box size of 20% of the controller's bounding volume.

5.10.3 Editing Gestures

Once recorded, gestures may also be edited. Properties such as position, orientation, and bounding volume extents can be modified on a per-waypoint basis, by either selecting them in the 3D view, or from the waypoint list pane on the right side of the application; it is permissible to select multiple waypoints, with the on-screen controls manipulating the properties of all selected waypoints. Additionally, waypoints can be added or deleted from streams at will. It is also possible to add and remove Speed and Synchrony groups to the gesture; once created, waypoints may be added to them by selecting the waypoint, and selecting the group from the drop down menu. Selecting a group will display a pane allowing properties such as jerk threshold and speed time limit to be set, as well as the group's name.

5.10.4 Saving and Loading Gestures

Once a gesture has been recorded and modified to satisfaction, it can be saved out for later reuse, either for further editing, or for gesture testing within a game linking against the MotionLibrary. Due to its ease of use and near ubiquity as a data

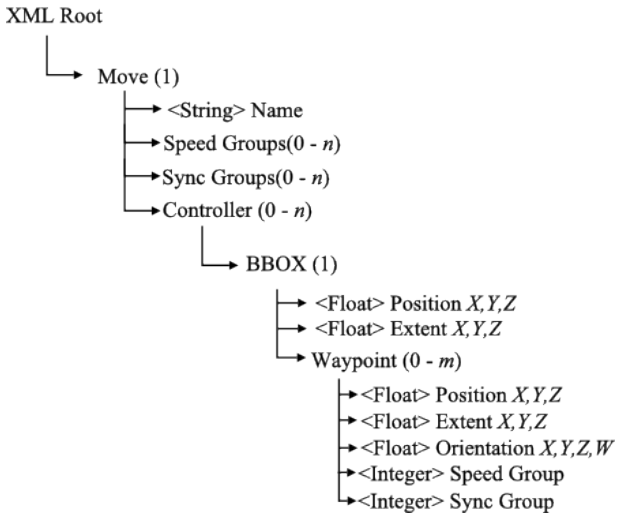


Figure 5.13: Hierarchy of data when storing recording movements, as represented in XML format

storage format, XML has been chosen as the default gesture storage mechanism. The nested hierarchical structure of the XML format lends itself particularly well to storing gesture data: 0-x motion controllers contain 0-y waypoints, each with 0-z components, leading to an XML hierarchy model as shown in Figure 5.13.

5.11 MotionBuilder: Rehabilitation Routine Creation

A gesture on its own is of limited use - simply performing the same move over and over is repetitive, and is too restrictive in terms of applying gameplay mechanics to when integrating into a rehabilitative game. There needs to be a method of choosing a range of gestures to attempt without resulting in a repetitive gaming experience, but which also gently increases the challenge presented to the player over time, by offering them moves relevant to their rehabilitative needs that have been determined to be more difficult to complete.

To this end, an additional tool has been developed for use in conjunction with gestures created by the MotionMaker program, which groups related gestures together into 'routines', which can be requested by game using the MotionInterface to use as the source of a pool of gestures for a particular game segment. This *MotionBuilder* tool generates and modifies a 'game file', which contain three lists of data, which together enable this ability: a flat list of gestures generated from the MotionMaker tool, a list of move *sets* which group moves together, and a list of *routines*, each of which references a number of gestures and sets to build up an entire rehabilitation exercise for use as input into a game. As with the gestures created by the MotionMaker, the game files are stored using the XML file format, and can be loaded into the MotionInterface by providing it with a file name upon initialisation.

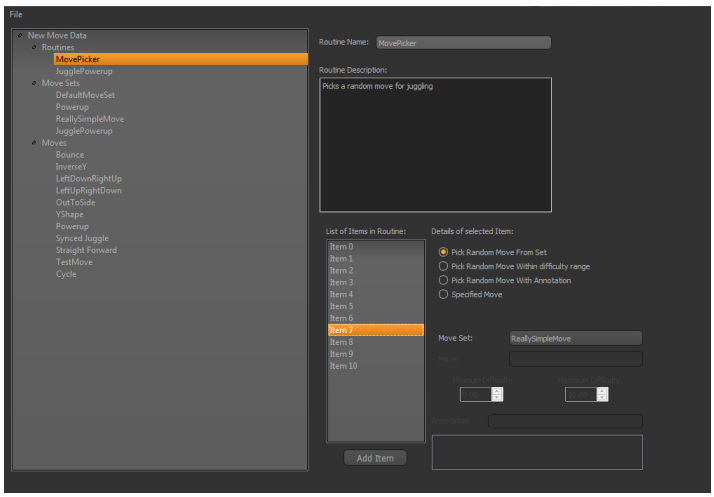


Figure 5.14: The *MotionBuilder* generation tool, allowing therapists to import previously recorded moves, annotate them as necessary, and add them to a number of routines, which can be loaded into games using the motion interface library

5.11.1 Gesture Data

The MotionBuilder tool allows a flat list of gestures created with the gesture creation tool to be imported, and allows these gestures to be tagged with a number of new variables, including a textual description of the gesture and its rehabilitative purpose, a number of text annotations, and a difficulty rating, saved as a real number between 0.0 and 1.0. Annotations are simple text strings which can be added, changed, and removed from a particular gesture within the the MotionBuilder application, and allow for related gestures to have their intent described in a textual manner. The difficulty of a move is intended to be used as a simple method of creating a relative ordering of moves from simple to hard to allow applications to pick appropriate moves, and is not intended to be an absolute metric of motion difficulty.

5.11.2 Move Sets

Moves can be further grouped into a number of named sets, each of which can have an additional textual description. This can be used to collate related gestures, such as those created to help with a specific ailment: A game for upper limb therapy could have a set of gestures validated for use for frozen shoulder, for example. Each individual move can appear in multiple sets within a game file if deemed necessary. Each game file may have an unlimited number of sets contained within it.

5.11.2.1 Routines

The primary purpose of the tool is to create a number of routines, each of which encapsulates the logic required to create an engaging rehabilitative experience when undertaken within a video game using the system. A routine consists of a name, description, and a number of entries, each of which chooses a gesture for the player to

attempt. How a routine entry chooses a gesture to provide to the game is determined by a selectable rule, which can be one of the following:

- Pick a specific move
- Pick a move randomly from a specified Set
- Pick a move within a specified difficulty bounds from a specific Set
- Pick a move with a specified annotation from a specific Set

This allows for a highly configurable routine, ranging from a set number of repetitions of the same move, to a wide range of differing moves with incrementing difficulty. As moves can be selected based on their annotations, it is possible to generate very specific routines with ease, to aid with physical rehabilitation of many problems; particular muscle groups or coordination issues could be added as annotations or within sets to later be selected from, creating medically applicable routines for a range of issues without any additional user interface requirements.

5.12 Threading and Message Pipelines

The underlying MotionLibrary that forms the basis for the rest of the system has been designed in a multithreaded manner, utilising asynchronous communication where necessary. Figure 5.15 shows the threading model employed by the system when it is used through the MotionInterface library.

The *Input* thread is used to constantly poll the state of the active motion device, and provide the rest of the system with the most up to date information on the device position, orientation, and velocity. The thread itself is set up and destroyed by the system thread whenever a new motion device is selected, and will execute constantly. When the Input thread receives new information from whatever underlying code interacts with the motion device, the internal state of the classes representing the device to the rest of the system are atomically updated using the appropriate C++ data structures to maintain consistency.

The *Detector* thread performs the process of determining whether a gesture has been completed correctly. The thread is started and destroyed by the System thread upon program startup, and while active repeatedly executes an Update function on an instance of a MoveDetector class; communication into and out of the thread is via the public interface of this class. This class allows for starting the gesture detection, taking in a specific gesture as a parameter. To asynchronously communicate the current progress through the gesture attempt, the MoveDetector maintains

a message queue. Messages include notification of a gesture attempt start being detected, a move being failed, and a move being completed along with the score achieved. These messages can then be popped off the queue by the system thread as necessary.

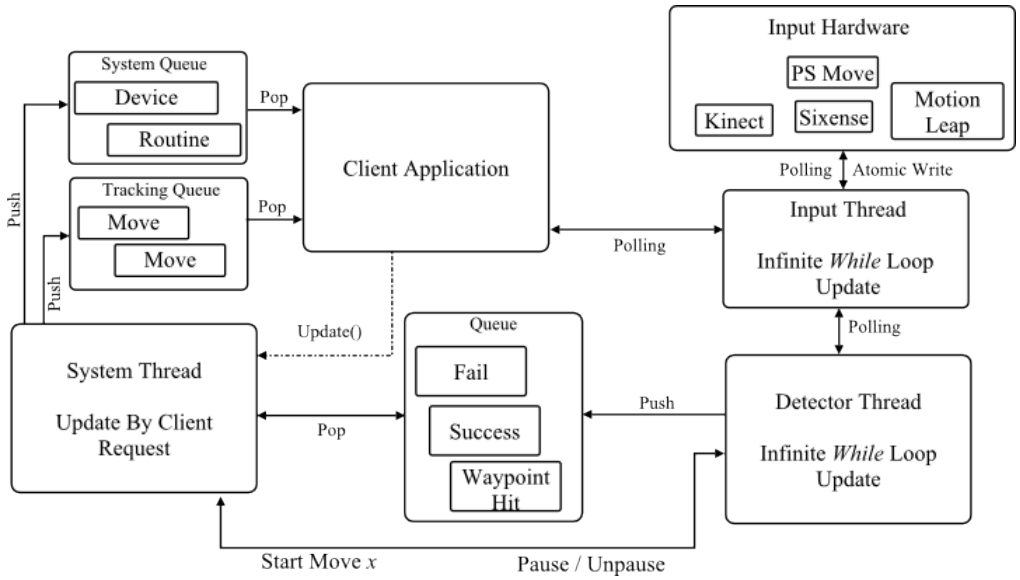


Figure 5.15: Threading and messaging model of the system: Queues are used extensively to allow for subsystems to be threaded and updated asynchronously

The *System* thread controls the overall state of the system, and provides bidirectional communication with the program using the library. Its main purpose is to execute the code within the Motion System system, as outlined in Section 5.9. To communicate with the external program, the System thread also maintains communication queues. Messages from the gesture detector are pushed onto the Tracking queue, while status information on the overall system itself, such as whether motion devices have been turned off, or whether loading gestures, routines, and moves has succeeded or failed, are pushed onto the System queue. This allows the client application to pop off and respond to enqueued messages at their own update rate, without impacting on the ability for the rest of the system to perform its actions in a timely manner.

5.13 Integration Example: Circus Challenge and Unity

To ensure the system's ability to be successfully used with an existing game, work was undertaken to integrate it into a version of the game *Circus Challenge*, used for research into stroke rehabilitation games as part of a project between the School of Computing Science and the Institute of Neuroscience at Newcastle University. The game is built upon version 3 of *Unity* (www.unity3d.com; *Unity Technologies*), a popular game engine used for numerous games across many different genres.

5.13.1 Unity As a Game Engine

The Unity engine allows for games to be designed and built within a fully-featured graphical user interface, that allows for the creation of *scenes*; grouped sets of game entities, arranged in a hierarchical fashion. The engine provides a wide range of capabilities to the entities within the game, allowing them to have advanced 3D graphics and audio, with physics and collision detection provided via Nvidia's PhysX library (www.nvidia.com; *NVIDIA Corporation*).

The primary language used for the development of games within Unity is C#, and much of the functionality of the engine is exposed to the user via C# classes. Rather than a deep inheritance hierarchy defining the properties of each game object, Unity game objects have their logic composed via an *Entity-Component* model: each entity is simply a container for a number of classes derived from a *MonoBehaviour* base class, which has virtual functions that are called upon certain engine events, such as when an entity is first added to the scene, or upon a new frame update.

5.13.2 Plugin Integration

The Unity engine also supports *plugins*, which are automatically loaded into the engine at run time, providing the dynamic library files are placed correctly within the engine's file structure. The C# language supports the calling of external functions that have been linked and exported correctly, complete with parameters and return values. This allows new functionality in dynamic libraries to be represented within Unity as an entity, composed with a MonoBehaviour derived class controlling access to the dynamic library.

To ease the integration of the Motion Control System into Unity, a file containing a C# class derived from the Unity *MonoBehaviour* base class is automatically generated when the MotionInterface is compiled from source, as detailed in Section 5.9. This class imports all of the public functions, and includes all of the structs and enumerated types necessary for correct communication with the system.

Where necessary, the datatypes used by the Motion Control System are automatically converted into Unity equivalents: positions and orientations are transmitted as a number of floating point values, and transformed into the Unity *Vector3* and *Quaternion* classes using 'shim' functions that sit between the imported library function and the Unity codebase. A similar shim function is used to transmit strings; text data is sent as a pointer to a null terminated character array, which is then copied and transformed into a C# string using the language's marshalling interoperability functions.

5.13.2.1 C Linkage Generation Details

To aid in the conversion process from C++ to C and C#, the composition of the C++ Motion System class has some limitations placed upon it. The main such limitation is that public functions may only use fundamental types for variables, or structs made up of only fundamental types. For example, the function to return the position of a controller does not return a copy of a Vector3 struct as it is not a fundamental type; instead it takes in three floating point variables by reference as parameters, and sets them to the x,y,and z coordinates of the position instead. This affords greater interoperability with other languages as it only assume that near ubiquitous datatypes such as integers, booleans, and floating point exist, and also avoids any confusion with data structure design choices; it is popular for data structures representing a 3 component vector to actually have 4 components, so as to allow greater cache-line alignment and more efficient loading of the resulting 128-bit section of memory into SIMD instruction registers.

5.13.2.2 C# Class Generation Details

The C# class file is generated as part of a post-build event of the MotionInterface class. An external program is executed, taking in the class header file of the Motion System C++ class, and an initial template text file, containing the skeleton of a MonoBehaviour class. The template text file is used to automatically integrate any functions or package imports without needing to modify the generation program itself. Using this as a base the program builds up the contents of a C# class, by scanning the class header file for the following data structures:

- Enums
- Structs
- Public functions within the MotionSystem class

Enum structures are copied verbatim into the class, as their syntax is identical between both languages. Structs with member variables compatible with C# are also copied over, with the addition of their variables being explicitly made public where necessary. Public functions are processed by breaking them down into three components:

- Function Name
- Function return type variable
- Function parameter variables

This allows the function to be reconstituted as an import declaration within the C# class file. For example, the example C++ member function:

```
int GetNumDevices();
```

Would be automatically expanded into the following C# statements:

```
[DllImport("MotionInterface" ,
CallingConvention = CallingConvention.Cdecl,
EntryPoint = "GetNumDevices")]
protected static extern int GetNumDevices();
```

The **dllImport** attribute is a C# language feature, that defines a function to be imported from an external library. It takes a number of parameters: a library name to import the function from, the convention to use to access the function, and the name of the function to import. As the MotionInterface library exports all functions using C-style linkage, the **Cdecl** parameter is used for the linkage type. The function declaration immediately following the **dllImport** attribute declaration will be mapped to the library function, making calling external library functions seamless. As the C# class is designed to be used as a base class for an encapsulated system, the access rights of the C++ class functions are changed from **public** to **protected**.

For functions which will be transformed by the C linkage creation process such that they take in references rather than return a non-fundamental type, the original function can be reconstructed within the generated C# class if an appropriate Unity data type is present. The program can transform Vector3, Quaternion, and the AABB (representing an *axis aligned bounding box*) data structures into their Unity equivalents.

This is easiest to see with a the example, in which a the C++ function *GetNormalisedPosition* is converted to C linkage, and then to a C# function. It has the following declaration:

```
glm::vec3 GetNormalisedPosition(int controllerNum);
```

As the Vector3 class is not a fundamental datatype, the function will be transformed to the following function when the C linkage is generated:

```
DllExport void GetNormalisedPosition(,
int controllerNum, float& x, float& y, float& z);
```

This will allow the position to be read into any C# program. To facilitate integration with Unity, these functions will be imported, and then a further C# function generated to turn it into a Unity Vector3 data structure:

```
[DllImport("MotionInterface" ,
CallingConvention = CallingConvention.Cdecl,
EntryPoint = "GetNormalisedPosition")]

private static extern bool GetNormalisedPosition(
int controllerNum, out float x, out float y, out float z);
```

```
public Vector3 GetNormalisedPosition(int controllerNum) {
Vector3 v = new Vector3();
GetNormalisedPosition(controllerNum , out v.x, out v.y, out
v.z);
return v;
}
```

Note that in this case, the imported function is made *private*, and the newly generated function *protected*, making the conversion 'seamless' to the rest of the codebase, and the API likeness maintained. By encapsulating all of code transformation within a single program executed at compile time, the process of adding functionality to the MotionLibrary was made straightforward, as changes could immediately be used by both the MotionMaker and MotionBuilder programs, and any Unity projects that were to utilise the library.

5.13.3 Gameplay Integration

To integrate the Motion Control System into a version of the Circus Challenge game, the *MotionInterfaceBase* C# class created as in Section 5.13.2.2 is further derived from, with a class designed to control the selection of gestures as necessary

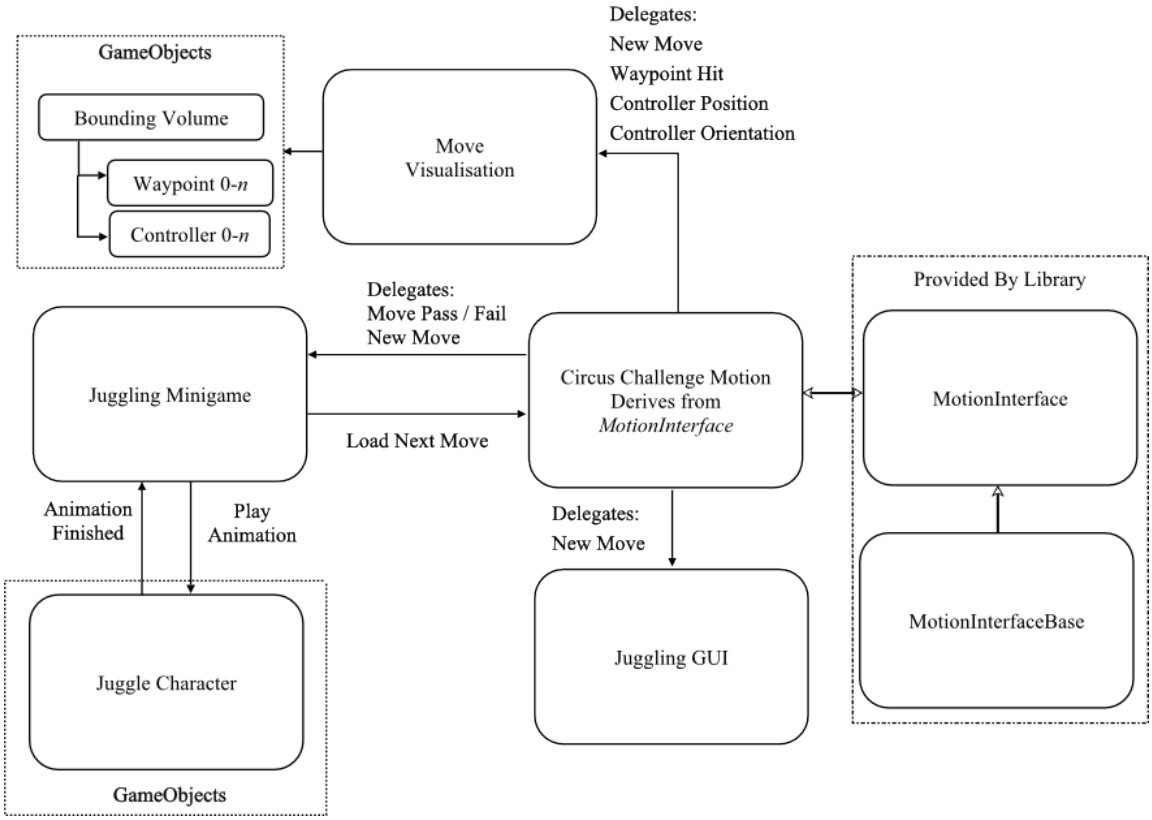


Figure 5.16: Circus Challenge Class Communication: The existing juggling minigame unity class was modified to register delegate functions with a new unity class derived from the automatically generated *MotionInterface* class, allowing integration of the motion system without extensive changes throughout the code

to allow the game to operate correctly. This class contained the code necessary to pop messages from the *MotionInterface* queues, and respond to them appropriately. The *Observer* design pattern is used to inform other entities within the scene of relevant information, via the usage of a number of *delegates* - classes can register functions to lists of functions within the MotionInterface class, that will be called upon events occurring, such as moves being started or failed. This produced the class communication shown shown in Figure 5.16.

The original Circus Challenge game had an on-screen character which performed the gestures that the player is required to complete, to help the player understand how to perform the next gesture. This relied on a number of prerecorded animations mimicking the movements hard coded into the game. To replicate this on-screen help, the gesture state including waypoint positions, volumes, and controller positions were queried, and visualised on screen using Unity entities, that were created and destroyed as the current moved changed. An example of this visualisation is shown in Figure 5.17. The waypoints of each stream are child objects of a parent bounding volume, which scales the volumes as appropriate to be visible on screen. Controllers are represented by coloured boxes that have their position updated each frame to represent the most up-to-date information from the motion system. The state of waypoints can be queried within the system, enabling 'passed' waypoints to be coloured differently.

The integration was successful, allowing a limited version of the game to run using the motion control system: the Juggling gametype was modified to hook into the system rather than the state machine based detection otherwise used, and could request a number of basic gestures from a test game file loaded into the system upon the start of execution, driving the character animation upon gesture success or failure using delegates as necessary.

5.14 Integration Example: Games developed by Coatsink and Nosebleed Interactive

The Motion System was used by two local game studios, Coatsink and Nosebleed Interactive, to develop upper limb rehabilitation games for the PC and Android platforms. Both used the Unity game engine, and were therefore provided with the auto generated C# class files, as well as an example Unity project derived from the work in Section 5.13.3, along with the required dynamic libraries to allow the Unity engine to operate with the system.

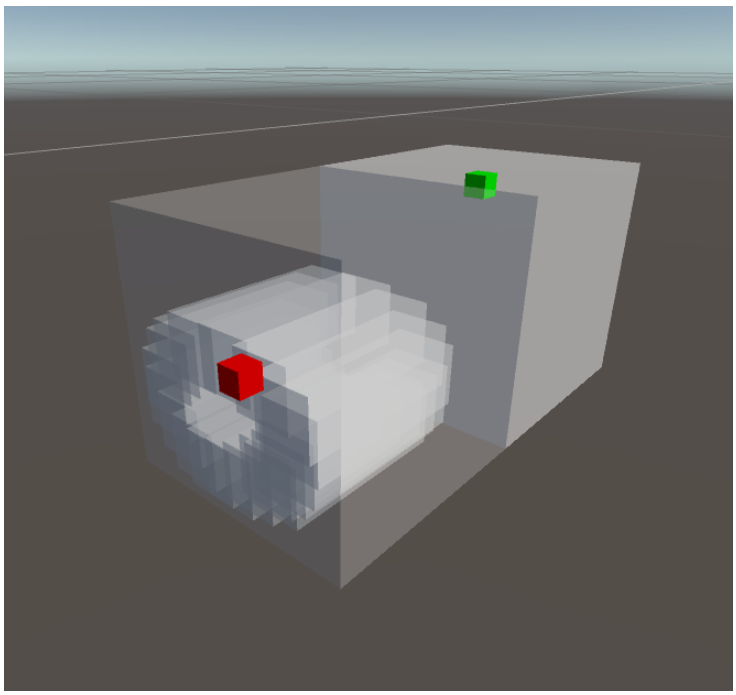


Figure 5.17: Gesture Detection Visualisation in Unity: Coloured boxes represent current controller positions, with grey boxes representing the normalised waypoints of the current gesture

The gameplay mechanics of both games are similar, with the player character progressing through the game automatically until faced with a challenge, whereby the player must intervene by successfully completing the indicated gesture. In *Sir Bramble Hatterson* (Figure 5.18: top), the player follows the adventures of a wizard, who comes across enemies on his travels, who must be defeated by depleting their hitpoints by casting spells at them, with the amount of reduction being related to the score metric of the on-screen move the player must perform to cast the spell. In Coatsink's *Skiing* game, the player helps a downhill skier avoid obstacles by completing the onscreen move when indicated.

Both games utilised the observer design outlined in Section 5.13.3 to hook the system into their own code. In the case of the Coatsink game, it was found that it was easier to treat the Motion Control System as a 'slave system' that could be turned on or off as necessary, so as not to 'interrupt' the game presentation with a request for a new gesture attempt. This led to changes to the internal mechanism of the Motion Control System to allow it to be set it into either 'automatic' or 'single move' mode, changing whether the system will automatically load in the next move upon completion, or will instead wait until indicated by the host application. The developers also expressed a desire to encode bonus 'victory' moves into the gameplay, whereby performing a number of moves well made the next presented move a simple 'cheering' or arms up gesture, to promote a positive feeling of achievement after a good performance. This was achieved by utilising the text annotation method described

in Section 5.11.2.1 to place specific celebratory moves into a routine, that could be searched for by the game to select as necessary.

The annotation system was again used by both software teams as a means of storing filenames associated with each gesture, relating to video files and unity animation files, that could be loaded into the games to assist with learning how to complete a move. Based on feedback on this usage, the MoveBuilder program was expanded to allow for files for video and animation to be directly linked to a movement, using a file dialogue, with support for basic video playback for verification purposes.

5.14.1 Android Considerations

The Android operating system is becoming a popular platform choice, not just for phones, but increasingly in tablets and set top boxes. Many such boxes, such as the Razer Forge TV gaming device (www.razerzone.com; *Razer Inc.*), include USB ports, opening up the opportunity to use low-cost computing hardware to interface with motion devices. Also, an increasing number of phones support the ability act as USB hosts, commonly referred to as 'USB on the go', allowing them to interact with other devices - the LEAP Motion finger tracking system has recently added support for this configuration, suggesting that such a combination could be for suitable for physical therapy of the fingers. For this reason work was undertaken to ensure that the Motion System works correctly on Android devices, so that as support for interaction devices comes to the operating system, they can be integrated and used. The games developed by both Coatsink and Nosebleed Interactive were both designed with the intention of running on Android, taking into consideration the size of the screen when creating the user interface, so as to remain legible when viewed from approximately arm's length.

Applications developed for the Android OS are packaged up into a singe file that contains all of the assets and files required by the program to function correctly, archived using the *zip* format. This means that unlike the PC platform, the game file and associated gesture files are not readily accessible via the standard C++ file stream interface. To allow the library to correctly execute on Android, the *zlib* (www.zlib.net; *Jean-loup Gailly, Mark Adler*) open source library was utilised within the MotionLibrary project, allowing read access to the apk file, the location of which is determined within the C# side of the application, and forwarded to the library upon program startup via a new function, **SetExternalResourcePath**. The apk can then be be loaded, and individual files unpacked into memory as replicas of the original, uncompressed files. The TinyXML library natively supports treating data in memory as an XML document, allowing compressed file support to be added with relative ease.

Figure 5.18: Games Developed Utilising the MotionSystem: *Sir Bramble Hatterson* by Nosebleed Interactive (Top), and *Skiing* by Coatsink (Bottom)

5.15 Feedback Discussion

Feedback on the motion control system was received at an event showcasing the various aspects of the system, with computer setups to demonstrate both the creation of new moves within the MoveMaker software, and of previously created moves being loaded into and assessed by the demo version of Circus Challenge. The intended audience of the event was both software engineers and healthcare professionals from the NHS, with the intended aim of putting forth the idea of increased collaboration between such parties to improve the quality of software delivered physical therapies. Feedback from the event in relation to the system was positive, with a number of people stating their desire for an easy to use method of creating customised therapies, as while people are aware of the concept of video games for promoting healthy lifestyles, primarily through the popular Wii Sports package, there was no way of recommending a product that could match someone's unique needs. Hands-on demonstrations of the MoveMaker software to physical therapists showed that the concepts behind the creation of simple gestures, including waypoints and the need to normalise them to a bounding volume of reachability, were easily understood, and it was seen as intuitive as to how to translate their current therapies to digital counterparts using the system and provided Sixense motion controllers.

5.16 Detection Validation Experiment

To assess the ability for the motion control system to detect user movements accurately, and with enough sensitivity to detect changes in ability over time, an experiment was performed. This took the form of a trial whereby a group of users played played a modified version of the *Circus Challenge* game across 3 sessions, with each session at least 2 days apart. In each session, the game presents to the player an identical set of carefully chosen movements to perform, with each individual gesture showing up multiple times in the set. All trial sessions were undertaken with the same motion sensing device, a Sixense TrueMotion setup consisting of 3 controllers - one to be placed in each hand, and one to be placed in a pocket, with the third device's reported position used as an 'offset' to the other two, allowing the user some degree of movement without affecting the reported position of the user's hands.

5.16.1 Experiment Setup

The experiment is based around the modified version of *Circus Challenge* outlined in section 5.13. The players in each trial group are tasked with playing a modified version of the TeeterBoard game type, in which the player must complete gestures in order for two acrobats to keep successfully bouncing on their teeter board setup - completed gestures will result in the acrobats performing tricks such as backflips and twirls, while failing movements will result in the acrobats falling off. For the purposes of this trial, the 'failure' state has been disabled - in usual gameplay, you are penalised by having to wait for the acrobats to return to the teeter board, inhibiting the maximum score possible within the time limit of the level, but as this trial is not concerned with the in-game scoring, it was deemed not necessary; this also ensures that the trial is consistent across all attempts.

Moves are presented to participants in a fixed order, such that each of the chosen moves is seen 3 times in total. This was achieved by creating a customised movement file and routine using the *MoveMaker* application. In addition to the trial moves, the players are asked to perform a 'arms by side' gesture between each movement, negating the impact that the previous movement's final position may have on the newly presented movement. In order to gain some valuable data as to the effectiveness of the software at dealing with players who suffer from limb weakness on one side, and thus generating asymmetric bounding volumes for their limbs, for the third and final trial players were asked to not outstretch their left hand when generating the initial calibration volume that gestures are normalised against, but instead keep their elbow crooked, and tense their shoulder, in a simulation of some degree of paralysis in the limb.

5.16.2 Trial Groups

For the experiment, a small cohort of volunteers was gathered, ranging from a young female of approximately 5'2" in height, to a middle aged male of approximately 6'5". All participants were neurologically intact, however one participant had recently suffered a fractured arm, resulting in limited motion in one arm; for the purposes of the trial, this was beneficial, as it allowed for an more accurate assessment of the ability for the normalised bounding boxes to accommodate asymmetry in the player's limbs, beyond the simulated asymmetry applied in the third trial.

5.16.3 Assessed Movements

As the motion control system has been designed to detect a number of different user metrics, the movements to include within the trial must be carefully considered so as to cover a wide range of use cases, including movements where orientation, speed, and synchrony are important.

The final selection of assessed movements consisted of 12 movements, each chosen to test different aspects of the motion detection system. Many of the moves were derived from the therapist-created gestures already present within the *Circus Challenge* game, as a demonstration of the system's ability to represent the physical actions a therapist might create for their patients. The chosen gestures are illustrated in Figure 5.19, and are as follows:

1. **Down By Sides**: Players must place their arms down by their sides. Tests bounding volume and synchrony.

2. **Y - Shape**: Player must extend their arm up and outward. Tests detection of y-axis movement, correct determination of bounding volume extent.

3. **Rowing**: Player must simulate a rowing motion with both hands. Tests accuracy of primarily z-axis movement and synchrony.

4. **Left / Right Balance**: Player must simulate reaching up with one arm, keeping other steady. Tests accuracy of positional scoring.

5. **Forward Hand Roll**: Player must move their hands in circles at a certain speed. Tests accuracy of speed scoring.

6. **Tilting**: Player must alternately extend one arm outwards and down, and the other outwards and up. Tests synchrony and xy-axis movement.

7. **Bicep Curl**: Players must curl both of their hands towards their shoulders at the same time. Tests orientation and synchrony.

8. **Chopping**: Players must alternately chop with their left and right hand, as if controllers were knives. Tests orientation and synchrony.

9. **Drumming**: Players must alternatively strike their hands down, as if beating a drum. Tests orientation and synchrony.

10. **Left Forward + Right Up / Right Forward + Left Up**: Players must extend one arm straight forwards, and one arm straight up. Tests bounding volume extents and synchrony.

Each of the movements selected is relatively straightforward to explain; additionally at the start of each trial session, participants are shown how to complete each of the gestures correctly in order to reduce the chances of mistakes being made during movement execution, while also accommodating for the lack of graphical instruction on screen, such as the 'John Doe' character in Circus Challenge. The waypoint visualisation shown in Figure 5.17 is shown on screen, to assist in determining what the root of any move completion issues may be.

5.16.4 Recorded Measurements

To analyse participant ability in the game, the *MotionLibrary* was enhanced with the ability to record gesture attempts to file, using the XML file format. This required some changes to the developed Circus Challenge code, in order to pass on the current user's name to the *MotionLibrary*, where it is used along with the date and time of each new gesture attempt to create a unique file name for each recorded attempt that the user makes. The recording file stores the position and orientation of each of the controller for every update that the detection thread makes, as well as storing each of the messages emitted by the system to the host application - thus, the frame in which a waypoint is hit, a gesture is started or reset, and if a gesture is completed is all stored in the file, along with the scoring for each of the gesture metrics described in Section 5.7.6.

To process the resulting recordings an application was developed, utilising C++, OpenGL, and the Qt windowing library. This application allowed gesture attempts to be visualised, or folders of gesture recordings to be processed, and statistics generated from them. For the visualisation, a single recording file could be selected and loaded, allowing the gesture waypoints to be loaded and shown in an OpenGL window, along with controls to allow frame-by-frame and realtime playback of the user's movements in the attempt. This allowed unusual or unexpected scores to be further analysed in an intuitive manner, allowing problems with the system or individual gestures to be found and dealt with. The statistical analysis function processed an entire folder's worth of gesture recordings, and allowing output of the

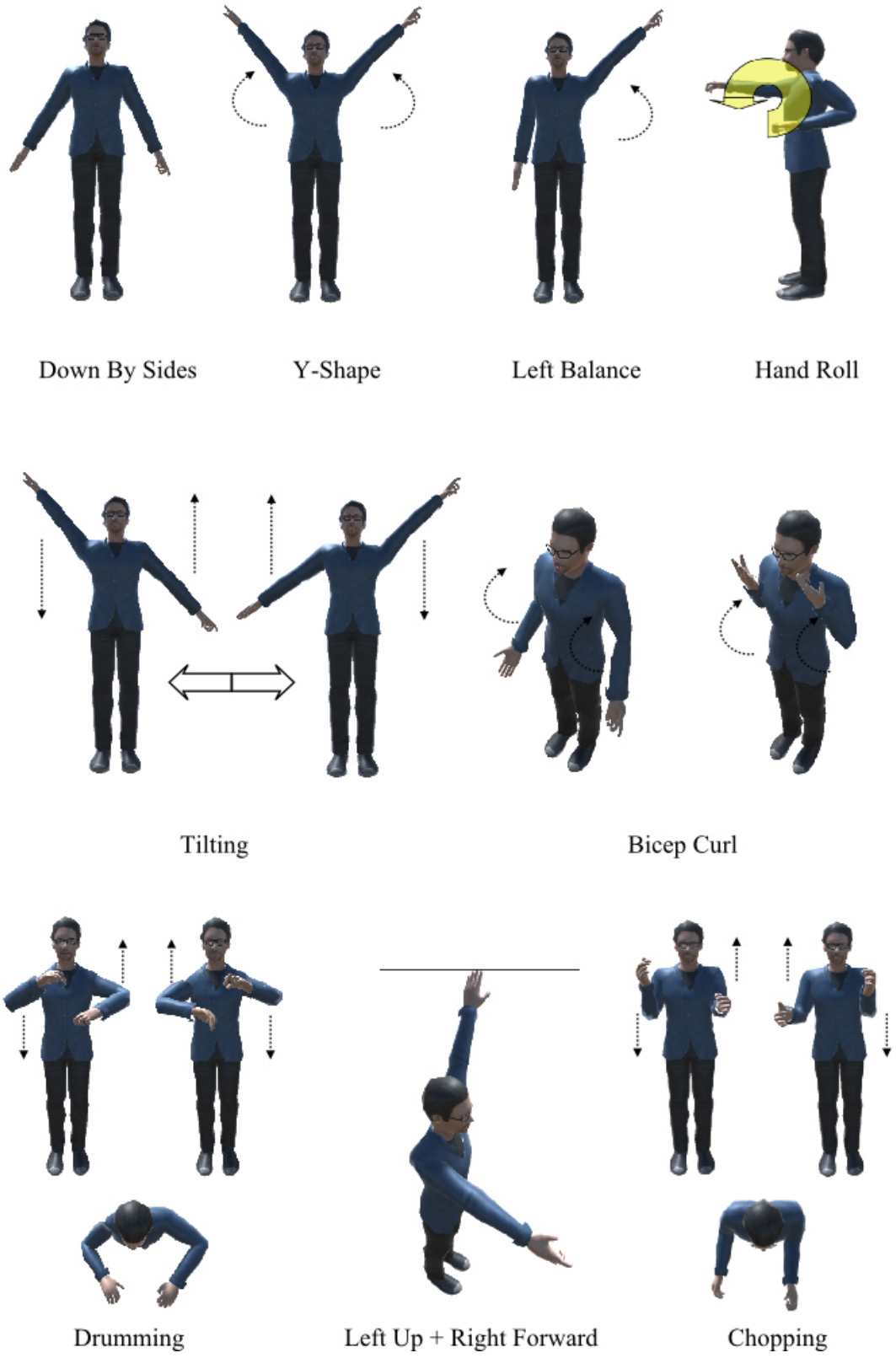


Figure 5.19: Games Developed Utilising the MotionSystem: *Sir Bramble Hatterson* by Nosebleed Interactive (Top), and *Skiing* by Coatsink (Bottom)

	Shape		Time	
Trial	Mean	Std.Dev	Mean	Std. Dev
Able Bodied A	0.7512	0.0540	6.0916	2.3762
Able Bodied B	0.7832	0.0491	5.5380	1.6520
Impaired	0.7274	0.0696	8.8407	2.8413

Table 5.1: Shape scores and gesture completion timing. Shape scores are between 0.0 and 1.0, and time is measured in seconds

player name, gesture, scores, and time taken to be outputted to a text file in CSV format. Statistical measures such as the mean and standard deviation for each user, each move, and each trial was also calculated using the program, allowing a wide range of useful data to be quickly derived from the recordings once the trial had completed.

5.16.5 Anticipated Outcomes

It is anticipated that there will be a trend towards an improvement in shape, orientation, and synchronisation scores for each gesture across all of the trials, as well as the time taken to complete the moves, due to the users becoming more familiar with the movements. The first time each movement is presented to them they are instructed as to how to complete them if necessary, and so it is expected that these will be the poorest, and slowest attempts. It is also anticipated that the final, faux-impaired trial will still result in good scores, despite the reduction in ability. A lower score than the able-bodied attempts is still anticipated, however, due to the fact that the movements are only simulated, and will not feel entirely natural to the user, which may also impact upon the time taken to complete gestures. As the participants are not impaired, it is expected that scores such as shape will in general be high, but any difference is score across the trials would be an indicator of the system's ability to detect even small changes in ability.

5.16.6 Results

All test participants completed all play sessions, with three sessions per participant, at least 2 days apart. In each session, every gesture was successfully completed, with all gesture attempts recorded and saved for parsing and analysis. At the start of each play session, users were explicitly instructed as to how best to complete each of the named gestures, with additional coaching in the first play session as to the correct manner in which the movement should be attempted.

5.16.6.1 Group Performance

The results of the trial are shown in Tables 5.1 and 5.2. As expected, users showed some improvement between able bodied trial sessions, most notably in time taken,

	Synchronisation		Orientation	
Trial	Mean	Std.Dev	Mean	Dev
Able Bodied A	0.2264	0.0751	0.2155	0.0540
Able Bodied B	0.2538	0.0380	0.2635	0.0425
Impaired	0.1985	0.0674	0.1810	0.0674

Table 5.2: Synchronisation and shape scoring, measured between 0.0 and 1.0

presumably due to participants being more comfortable and familiar with the gestures being performed. The faux impairment trial showed the longest time taken to complete moves, and also the most variance - some moves such as 'down by sides' can be completed fairly trivially , while the time taken to complete more complex gestures is exacerbated by the forced limitations on the left arm. Shape scoring too was inhibited by the forced impairment, showing the lowest mean score of 0.7274 across all trial participants (see Table 5.1), indicating that the detection and scoring method was sensitive enough to represent the lower ability displayed in the impairment trial session.

The scores generated for synchronisation and orientation shown in Table 5.2 continue the trend, with an improvement between trial session 1 and 2, and then a drop off for the impairment trial. The scores were lower than anticipated, but consistently so across all participants, indicating a common cause, discussed in Section 5.16.8.

5.16.6.2 Move Detection Performance

The mean scores generated for each individual movement are shown in Figure 5.20. As can be seen from the chart, most movement attempts resulted in a detection score of greater than 0.66 - the mean score across all gesture attempts was 0.759. The most interesting results are the lower than expected scores for drumming and rowing gestures, and the wide variance in the down by sides gesture, which has a standard deviation of 0.248, almost twice that of any other gesture.

The variance in the down by sides motion is almost certainly linked to the move's position in the trial routine - occurring after every other gesture. This means the user's limbs are likely to be in a variety of different positions upon the tracking beginning; for some, the arm will already be in the correct position from the previous move's completion attempt, yielding a high score, while in others the limb must be moved to the correct position. As the down by sides gesture has only one waypoint per limb, it will be completed as soon as the user's hand enters the waypoint, placing it at the edge of the scoring area - this was an unintended side-effect of the way in which scores are calculated that was not discovered until after the trial. For such gestures in the future, it would be better to add some form of heuristic for determining when a user has finished their final movement, allowing the final waypoint in

	Score		Time	
	Mean	Dev.	Mean	Dev.
Trial	0.759	0.105	6.73	4.49
Impaired	0.735	0.121	9.03	5.00

Table 5.3: Group versus impaired participant comparison. Score is measured between 0-1, time measured in seconds.

each limb's waypoint set to generate a score more accurately representing the user's motion. As the down by sides gesture was only designed for forcing the user to place their arms such that each gesture was attempted from the same starting pose, the gesture still served its purpose, despite the appearance of a wider variance in completion quality.

Inspection of attempts at performing the drumming and rowing gestures using the attempt visualisation program outlined in 5.16.4 suggests that the gestures were being performed incorrectly, due to a number of factors. For the drumming gesture, its similarity to the chopping gesture seems to have led people to hold the controllers at the wrong orientation, and too close to their middle, when the intention was for the arms to be outwards, with their controllers facing inwards, as if they were drumsticks beating on a drum. While the gesture was still successfully completed, the changes in arm positions of the users to the intended led to the score being lower than anticipated - this does at least suggest that the system's ability to score correctness was working, as the gesture visualisations indicated that the moves were clearly not as intended, but were similar enough be be viewed as a legitimately poor attempt at the movement. For the rowing gesture, several participants brought their arms in up to their chest for the simulated 'pull' of the oar, when the original recording brought the controller closer to being under the armpit. As with the drumming gesture, the positions were close enough to catch the recorded waypoints, but sufficiently distant enough to only generate relatively poor scores. The consistency of the ways in which the gestures were incorrectly completed raises an interesting area for improvement to the system - waypoints that are repeatedly completed with a poor score could perhaps have some contextual feedback added to them at the editing stage - a text message stating 'Bring arms in closer!' for example, so as to inform users as to how best to improve their gestures. This could perhaps also be augmented with some automatic generation of messages by the system, perhaps as a preset list of generic assistance messages that could be translated into new languages as necessary.

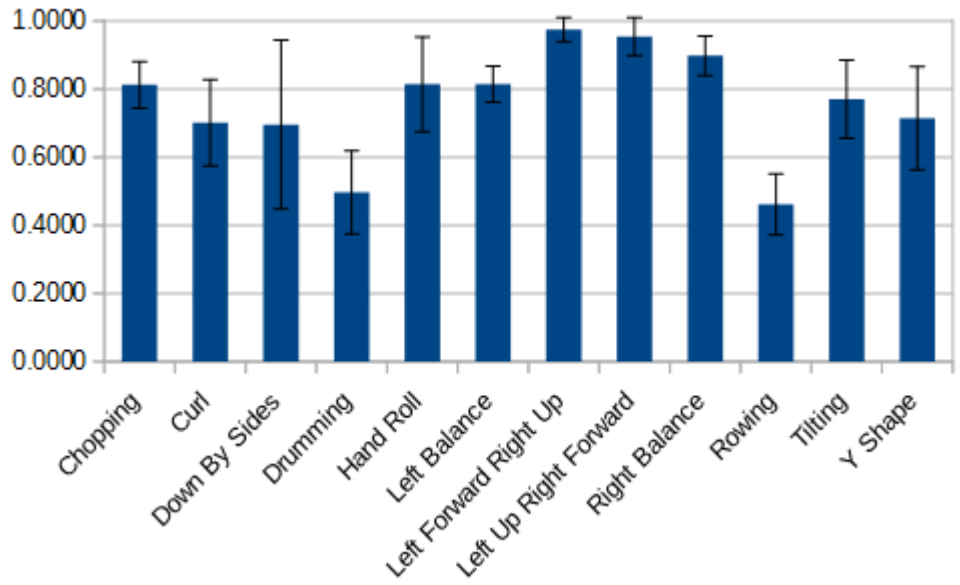


Figure 5.20: Per move results. Y-axis shows mean gesture detection result for each move.

5.16.7 Impairment Result

As the trial participants included one person undertaking physical rehabilitation due to an arm injury at the time of the trials, it is beneficial to look at their results in comparison to the group as a whole. Due to their injury, the range of motion in their right arm was inhibited, resulting in extremely asymmetric reachability bounding volumes, as evidenced in the visualisation in Figure 5.21, in which the right hand volume in green is noticeably smaller than the left hand in red. Despite this asymmetry, the participant was able to complete all moves, with score results similar to that of the rest of the participant cohort. A comparison between the two is shown in Table 5.3, showing that while the impaired participant was considerably slower than the group average at completing each gesture, their scores were comparable (a mean of 0.76 for the group versus 0.74 for the impaired participant). This is largely to be expected, as in this particular case the impaired participant had no specific muscle weakness or atrophy, with mostly only muscle pain limiting range of movement - within the area that could be reached, movement was free, and thus once the system had calibrated to the reduced reachability, the gestures themselves could be completed acceptably, albeit at a slower speed.

5.16.8 Experiment Discussion

While small in scale, the experiment does show that the motion control system is capable of detecting gestures requiring positional accuracy, and containing elements of synchrony and orientation as part of the determination of success of a particular move attempt. Trial participants were generally successful at matching the physical positions of the gestures, with a mean shape score of 0.7593, with a standard deviation of 0.105, indicating that the results are consistent across the group. As was

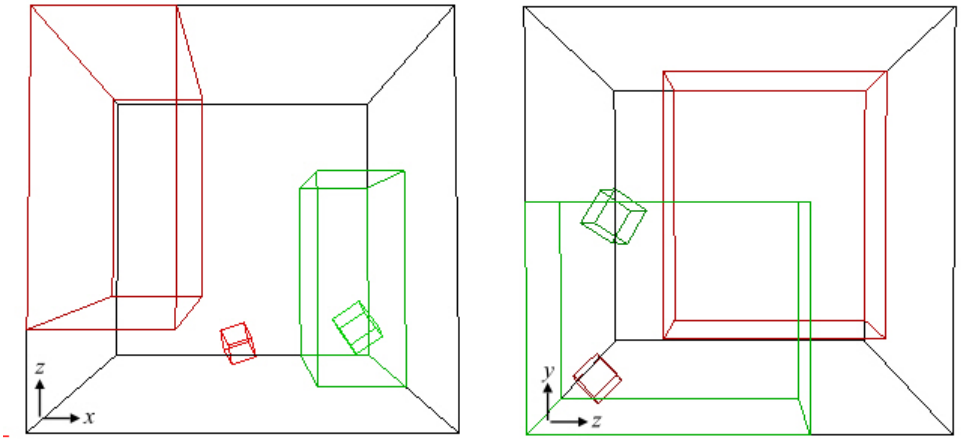


Figure 5.21: Left: A top-down view of bounding volume asymmetry in impaired participant. Right: A side-on view of same.

expected, the hit score improved over the course of the able-bodied sessions, and while lower in the impairment trial, still generated results that indicate the movements were being attempted correctly.

One issue across the trials was in participants adherence to the synchrony and orientation requirements in the movements. In part this may be due to the on-screen visualisation used in the trial, as shown in Figure 5.17. From discussion with participants post-trial, it was found that they were naturally inclined to try and 'complete' each waypoint in turn to fill in the visualisation, rather than naturally perform the gesture the visualisation represents. As the waypoint visualisations have no displayed orientation as such, being simple axis aligned bounding boxes, participants were completing the gestures without necessarily taking into consideration the orientation of the limbs specified in the pre-trial instructions. Orientation requirements for most waypoints in the tested gestures was to be within 90° of the recording, but it was too easy to be outside of this range for many waypoints, such as ones requiring full arm extension, where there is over 180° of rotation available across the arm joints while still meeting the position requirements, which without constant reminders as to the correct pose, can be too easy to 'complete' incorrectly.

The problems relating to gesture visualisation with could be alleviated by using videos of someone performing the required gesture to prompt players as to how to successfully complete it - this is not an ideal solution for wider-scale deployment of the system, however, as video files are large, and having to create a video for every movement a therapist creates for their patients is infeasible. A better solution for future gesture recordings may be to generate animations for a humanoid mesh using the gesture waypoint positions as keyframes, allowing automatic creation of onscreen visuals as new gestures are created.

5.17 Conclusion

This chapter has outlined the creation of a system whereby limb movements designed to form part of a physical rehabilitation routine can be described as a number of waypoints with a position and orientation in 3D space. The system is capable of grading the ability to recreate these movements using motion controllers, by determining which waypoints are passed through, with further optional constraints such as speed and smoothness of motion, and the level of synchrony between multiple motion controllers. A number of such gestures can be combined into 'routines' that act as interactive therapy sessions, presenting a new challenge to the player each time by randomly selecting from pools of gestures based on a difficulty range, or by a textual annotation match. This is achieved via a number of programs to facilitate the creation and editing of moves and routines, based on a common, cross platform library that can additionally be imported into C# for use with the *Unity* game engine, via automatic code generation techniques.

The work has been created to allow increased collaboration between medical professionals and game developers, by encapsulating the needs of both into a single package that can be integrated into existing game engines. This allows game developers to interface with a number of motion devices to detect 3D gestures, without requiring knowledge of the underlying health aspects that comprise the movements. Likewise, medical professionals can now develop bespoke physical therapies for their patients to be delivered via games and engaged with in the home, without requiring any knowledge of game design or development.

Preliminary results into the success of the system at meeting its goals are positive, with local game developers using the library to develop test games suitable for rehabilitation purposes, deployable on both the PC and Android platforms. Integration of the system with their work has been straightforward, and suggestions on how to facilitate their gameplay ideas has led to improved features and new functionality in the system. Feedback from physical therapists who have been exposed to the gesture and routine creation programs displays interest in the system, stating that it could provide a beneficial service to those requiring bespoke physical therapies.

While the accuracy trial was small scale, it shows that gesture attempts can be completed and graded using a combination of the gesture detection system and an appropriate video game. This suggests that with the correct development of the gesture detection system, and the creation of a larger library of movements to detect and grade, that the integration into games such as those noted in Section 5.14 would allow therapists to deliver to their patients a more engaging interactive therapy, that can be monitored for changes in ability over time.

Chapter 6

Conclusions and Future Work

6.1 Introduction

This chapter concludes the thesis, providing an overview of the work discussed in each chapter, the contributions the work provides, and a discussion of potential avenues for future work.

6.2 Contributions

The work described in this thesis provides a number of contributions to the field of video game delivered therapy, which together allow for a more informed and integrated development of games designed for physical rehabilitation in the home. Firstly, a method by which to determine the quality of data output from motion devices suitable for video game interaction is presented, and demonstrated by evaluating a number of modern devices. Such a measure can be used to evaluate the suitability of devices for interaction with therapies requiring a certain level of accuracy in positional and rotational movement. A number of force sensing devices are also investigated for their use in therapies requiring measurement of muscle strength, or for when holding conventional motion sensing devices is hard or impossible, such as with acute stroke.

The second contribution is to demonstrate the importance in how rehabilitative tasks are presented to a user. By developing a game suitable for rehabilitation of acute stroke using a bespoke, modifiable game framework, a number of scenarios can be explored. Work using the rehabilitation game has shown that in-game compound motor tasks are sensitive to how they are conveyed to the user - a multi-part task explicitly described as such results in lower in-game performance than if it was described in broad terms; not only was performance degraded, but seemed to plateau at a lower level, suggesting a permanent motor learning deficit. This suggests that care should be taken when integrating rehabilitation movements into a video game

designed to be used unsupervised, so as to not have a negative effect on a patient's ability to perform important compound motor tasks.

The final contribution is the development of a system that facilitates the creation of video games with a physical rehabilitation component. The system allows medical experts to create gestures in 3D space with complex interaction between limbs, which can then be attempted at a later date by patients, who will then be graded on their ability to replicate the movement. The system also eases the integration of rehabilitative moves into a video game by encapsulating the loading, testing, and grading of gestures, as well as the interaction with a range of movement detecting devices. This allows greater collaboration between medical experts and software developers, allowing more professional-looking, and more medically beneficial games to be developed.

6.3 Thesis Summary

Chapter 1 provided an introduction to the thesis. A broad overview of the field of rehabilitative gaming was described, to provide context to the work, along with the research contributions. The focus on hemiplegic stroke is introduced, leading to justifications of the work undertaken. A list of journal articles and conference proceedings where work described within the thesis has been published was also provided.

Chapter 2 provided a detailed investigation of the background and related work relating to the topic of rehabilitative gaming. This mainly focussed on academic literature describing uses of video game technology to assess or aid patients suffering from a wide range of illnesses, from obesity, to neurological conditions such as cerebral palsy and stroke. The research began with a look at cases where existing game technology, such as the Dance Dance Revolution arcade machine, has been used to encourage people to engage with exercise routines.

The related work continued by discussing ways in which more modern home console peripherals, such as the *Wii Remote*, *Wii Balance Board*, *PS Move*, and *Microsoft Kinect* have all been investigated for their applicability towards health gaming applications, finding a number of trials utilising them, with varying, but generally positive results.

The chapter continued on to discussing methods by which the severity of disability following stroke can be measured. Metrics such as the *Wolf Motor Function Assessment*, *Chedoke Arm and Hand Activity Inventory*, and *Fugl-Meyer Assessment* are

discussed. Such metrics form the standardised methods by which healthcare professionals assess their patients, and thus determine what medical interventions best suit their needs. The ability for a software package to be able to replicate the ability to measure such metrics is introduced, forming the core of the justification for the work undertaken, as such a capability would allow for more informed monitoring of remote teletherapy.

The chapter concluded with an investigation into the methods by which three-dimensional gesture detection was undertaken. Such methods are capable of detecting whether physical movements made with input devices such as the Wii Remote, PS Move and Kinect match up to specific desired gestures, potentially allowing for software employing them to be able to detect patient success at completion of their prescribed rehabilitation routines.

Chapter 3 introduced a method by which the quality of the output of motion tracking devices such as the PS Move and Kinect could be measured. A quality score, comprised of individual metrics that measure the ability of the device to report the correct position and orientation was described. The positional metric was formed by projecting positional information against a plane, and then determining the circularity of the resulting polygon using the area and circumference of the polygon. The orientation metric is constructed similarly, by projecting the unit length normal that represents the forward direction of the device against a plane, and determining circularity. This quality metric was used as the basis for a number of experiments, in which the PS Move, Wii Remote, Kinect, and Sixense devices were tested for their ability to record correct representations of human movement, at a number of distances. This movement comprised of a number of test participants rotating their outstretched limbs in the sagittal and coronal plane, and their whole body around the transverse plane; this provided a complete representation of the device's ability to detect motion in 3 dimensions, including cases of 'hard error' where the limb becomes obscured from the device.

The chapter then introduced an investigation into a number of force sensing devices; such devices being useful for determining a patient's ability to hold and apply pressure, and for operating as a means of primary input for those suffering acute symptoms from neurological conditions such as stroke. A number of commercial devices, including the Wii Balance Board, Saitek X-65F flight stick, C500 force sensing pads, and iLoad Mini force sensing devices were tested, for their performance in three metrics - their accuracy, their sensitivity to change, and their propensity for displaying signal bias and noise. Accuracy was determined across a number of different forces utilising weights, by measuring the difference between

the output in kilograms of a number of devices, and the weights themselves. Sensitivity was determined by determining how small a weight could be added to result in a perceptible difference in mean output, measured over time to reduce the influence of signal noise. Bias measurements were taken by measuring how device output changes over time, including the absolute change, and the sum of relative change of time, allowing both overall bias, and likelihood for signal noise, to be measured.

The chapter concludes each section with a discussion of the relative merits of each of the tested motion and force sensing devices, and describes possible applications for the devices relating to the advancement of stroke therapy, with focus on hemiplegic stroke.

Chapter 4 presented a toolkit for the creation of simple computer games, designed to be used with force sensing devices. The toolkit's purpose was to facilitate the creation of games to aid investigations into how best to enable video game interaction with patients suffering from acute stroke, as part of a wider research project undertaken with Newcastle University's Institute of Neuroscience.

The toolkit was designed to be highly adaptable to the requirements of research collaborators, through the use of configuration files that controlled almost every aspect of the games created upon it. The toolkit also supported the recording of data from a number of force sensing devices to a high fidelity, allowing the smallest of movements to be detected, and recorded to a file for further analysis.

A simple game was built using the toolkit to undertake collaborative research into inter-limb transfer of motor learning, and to investigate links between methods of task instruction, and their impact on motor learning over time. The game is played using pairs of force sensing controllers under each hand, with the player being tasked with moving an on-screen cursor quickly and accurately over a number of targets for a specified amount of time; this is presented to the player as controlling a spaceship to shoot down a number of incoming asteroids.

The investigation into instructional motor learning was facilitated by the the configurability of the toolkit, as it allowed rapid prototyping of different gameplay mechanisms, including the user interface and how information is conveyed to the user, as well as game specifics such as changing the number of asteroids to be destroyed, how long they must be 'locked on' to to be destroyed, the randomness of which hand to use, and modifications to how the targets moved on screen.

Research conducted using the toolkit found that the method by which an in-game task was related to the player had a significant impact on their ability to complete it; and that an incorrectly conveyed task would lead to a permanent reduction in performance, when compared to scores by people who had been correctly trained in how to complete the task. This was achieved by performing a test trial with two groups: group A was told explicitly to quickly move their in-game cursor to the target before attempting to track it, while group B was simply told to track the target as well as possible. This effectively presented the task in two different ways - one as a single motor task, and another as a compound action split in two. The results showed that the group who perceived the task as a compound action were significantly worse at the in-game metrics. The result is important, as a rehabilitative game designed to train and exercise specific moves will be of little benefit if the method buy which the task is conveyed actually damages the ability to correctly replicate them.

Chapter 5 detailed the creation of a set of tools that would enable the creation of interactive video games that utilise motion detection devices to detect whether the player was correctly engaging in the completion of a set of prescribed moves, or gestures. This work was undertaken with a view to enabling the creation of professionally developed video games that have rehabilitative gestures encoded within them driving the gameplay element. It was determined that the best way to enable this concept was to make the therapist driven aspect (gesture creation and therapy routine creation) completely separate from the video game developer driven aspect (the creation of the actual game). It was also deemed useful to abstract the interface between software and the motion controllers themselves, to ease development, and afford some measure of expandability to the system. This allows game developers to create games that can be of benefit for rehabilitative purposes, but without having to have in-house expertise in motion detection or physical rehabilitation.

A software library was developed, containing a system capable of reading positions and orientations from a number of motion devices, as well as saving and loading 'gestures' - streams of positions in 3D space that represent a specific movement. The system is capable of detecting when a gesture is attempted with a connected motion device, and will grade the attempt based on a number of metrics: shape, orientation, synchrony, speed, and smoothness. These metrics allow for a high detailed view of a patient's ability to complete gestures.

From this core library, a dynamic library suitable for use as a plugin in game engines is automatically generated, using code transformation techniques, allowing for any game capable of dynamically importing C-linkage functions to query motion devices, and receive updates as to moves attempted and completed.

To assist in the creation of gestures and rehabilitative therapies, the core library was integrated into two applications, designed to be used by occupational therapists. The first program, MoveMaker, allows gestures to be created in real-time using any of the supported motion devices. Created gestures can be saved, loaded, and modified as necessary to build a replication of the desired physical movement in 3D space. These gestures are represented within the system as a series of 'waypoints' for each controller taking part in the gesture, with a waypoint being made up of a 3D position and orientation, as well as a spatial volume to represent the tolerance to deviations when attempting to replicate the gesture. By determining whether later attempts at the gesture pass through these waypoints or not, it can be determined how successful the move is.

To allow previously recorded gestures to make up part of an overall rehabilitation therapy, the MoveMaker program was developed. This program allowed the collation of previously recorded gestures into 'sets', additionally being graded with a difficulty, and tagged with a number of text-based annotations. The program also allows the creation of 'routines', which are user-programmable sequences of gestures, which when attempted in sequence make up an individual therapy session. Such routines allow for each sequence element to be a gesture chosen either at random, from within a specific set, with a specified difficulty level, or with a specified annotation. The data from this MoveMaker program can then be saved out as an XML file, to be loaded in by games at a later time.

The system allows for bespoke gestures and rehabilitation routines to be created be experts on a per-patient basis as necessary, without the game developer being involved, potentially allowing for a video game to be deployed in the patients home, with progress stored for later examination by occupational therapists.

The chapter continues to describe preliminary work being undertaken within the games industry utilising the system. A number of established video game developers are shown using the system, employing it in Windows and Android based games built using the Unity game engine. Integration with the Unity game engine is described, as well as preliminary feedback from the video game developers as to the suitability of the system. Further feedback from a preliminary showing of the software to healthcare professionals is discussed, in which it is indicated that home delivered physical therapy is seen as desirable, and that the system is a welcome step towards bespoke physiotherapies.

A small trial to test the system's ability to detect and grade a user's attempt at replicating a number of 3D gestures is discussed. The trial, while small, and with able-bodied participants, indicates that the system is capable of determining how well a gesture was replicated, and graded users appropriately. Some small issues relating to user's not performing moves correctly were discovered, indicating that while the core mechanics of the system are capable, more work should be done to allow the generation of better on-screen instruction, and textual feedback as to movement success.

6.4 Thesis Discussion

This work investigates the means by which interactive software designed to aid in physical rehabilitation can be improved, with focus on software designed for use in the home without supervision. The background work shows that while there has been much research into rehabilitation games, these tend to be rather simplistic, with graphical and gameplay aspects that lack the level of 'polish' that a professionally developed game is expected to have. Many of the research projects have quite specific scope, making their results hard to apply to the wider field of rehabilitative software. Where research has been performed using commercial-quality games, such as *Dance Dance Revolution*, and *Wii Fit*, the games have not been modified from their publicly released state, and thus are not necessarily tailored to providing maximum benefit as a rehabilitative tool, but rather as an entertainment product with a health benefit as an aspirational goal.

It is clear from the background and related work that there is a scope to improve the methods by which a game with a therapeutic goal interacts with a patient. While the background research does indicate that video game based therapies are capable of having positive effects on a person's well-being, there is little work to suggest how best to improve game design such that the potential improvement is maximised. For the research undertaken into rehabilitative games to be fully realised as a product to deliver therapies to those that need it, it is clear that more commercial involvement from those with expertise in game design and development is required.

The work described in this thesis aims to help solve many of the issues. The work in chapter 4 presented an investigation into how minor changes in how the rehabilitative tasks encoded within a game are presented to the user can have significant impact on the potential for improving their performance at completing them over time. This suggests that it is not merely the presence of a therapeutic element within a game that leads to the improvements suggested as possible within the background work, but that specifics of integration play a role, too. The results from chapter 4

also indicate that therapy games coupled with force sensing devices can be sensitive enough to detect learning of motor skills in older post-stroke patients, opening many research avenues relating to the types of motor task that can be encoded within a game for such patients to engage with.

Chapter 5 provides a working example of how to ease collaboration between experts in the fields of physical rehabilitation and game development. By allowing game developers to target a library that can determine the player's ability to successfully replicate a given movement, the barrier to entry to collaboration on development of rehabilitative software is lowered. Similarly, allowing therapists to create new moves and routines, potentially on a per-patient basis, allows for truly bespoke therapies to be introduced into a patient's rehabilitative routine, widening the appeal for such a product.

Deployment of bespoke therapies in the patient's home requires careful consideration of the input device used to measure patient performance. To this end, there must be some method of determining the relative merits of these devices. This thesis proposes and details such a solution, via the measurement of positional and orientation tracking accuracy in a variety of conditions.

6.5 Future Work

There are a number of potential avenues for future expansions to the work in this thesis, both in the areas of gesture detection, and gameplay integration.

6.5.1 Rehabilitation Response Markers

The work within this thesis relating to the search for rehabilitative success indicators is only a first step towards understanding how best to integrate effective therapeutic actions within a video game environment. The Asteroid Game created as part of this thesis was limited to a single compound action, with only minor variations. There is room to research methods of integrating multiple movements into a video game delivered therapy session, and the impact upon motor learning multiple separate actions may have - it may be necessary to group certain types of movements together to reinforce motor learning, or slowly integrate different types of movements over the course of a therapy session.

There are more social considerations to developing an effective therapy game. Video games are commonly based around the concept of controlling some sort of on-screen character or object - the spaceship in the 'Asteroid Game' built upon the gaming toolkit is an example of this. There is potential research for determining

whether 'realistic' or 'cartoon' style characters engage people more with their therapy, or whether characters resembling their race, gender, or age impact engagement. Video games are often designed with multiple players in mind - either in a 'versus' setting as in the *Wii Sports* tennis game, but also increasingly in a collaborative role, such as the world building in *MineCraft*. It would be of interest to see how such multiplayer environments impact upon video game delivered physiotherapy - does the chance to compete against others improve one's engagement with the game, or is it better to allow the family to play together towards a common in-game goal, mixing traditional and motion tracking controls as necessary to allow a wide range of people to play.

6.5.2 3D Gesture Detection

The resurgence in popularity of virtual reality applications has led to the development of new body-tracking technology that may be of relevance to software delivered physical rehabilitation. Upcoming devices such as the Leap Motion, Sixense STEM system (www.sixense.com/wireless; *Sixense Entertainment, Inc.*), and the Pebbles interface finger tracking system recently purchased by Oculus (oculus.com; *Oculus VR, LLC*) are all possible candidates for inclusion in the Motion Tracking System developed in this thesis, allowing for greater fidelity of movement, and the ability to track more of the user's body in real time. To this end, work has begun on integrating the Leap Motion device, which in conjunction with a body tracking device like the Microsoft Kinect, could allow for a detailed recording of the user's upper limb. More thorough research will have to be done to determine whether the existing scoring metrics are suitable for finger-tracking gestures, however, and the detection of digit movements may prove to be a subtly different problem then that of detecting the upper limb as a whole. Collaboration with physical therapists will be required to determine how best to fully utilise hand tracking for therapeutic purposes.

To improve the detection quality and accuracy of scoring metrics against those of traditional gradings such as the Fugl-Meyer assessment, preliminary work is being undertaken to recreate the moves used for assessment within Circus Challenge using the Motion Control System. The large amount of data from recordings of the device positions and orientations taken during trials with the game could then be processed and fed into the motion control system, allowing a comparison to be made between the scores generated by the system and that of work undertaken with Circus Challenge itself. Experimentation with the data should allow improvements in the detection to be made, whether by the changing of the bounding volume of created waypoints, or in changing how components such as velocity and synchrony are weight into the final scoring.

While the waypoint system shown in this thesis has been shown to be an effective way of determining how well users are completing 3D gestures, they may not be the only solution. While the recording is intuitive, and results in an easy to use interface for understanding the basic spatial relationship between waypoints, the way in which waypoints must be tagged as requiring synchrony and speed is currently non-optimal. If a user is incapable of performing some part of a gesture, it can currently become 'impossible' to complete - for example if the user is required to lift their arm up such that their arm faces directly forward, and then to rotate the wrist, the waypoints of the gesture will be such that if the user is tiring and not able to fully lift their arm for this attempt, they won't reach the waypoint positions for the wrist rotation, even if they can perform that section of the movement at a slightly less elevated arm position. It may be necessary to split waypoints up into 'stages', with each stage's waypoints moving to the best position reached in the prior, and so allowing the movement to be completed. Such a change in the gesture would have to have the attempt marked as imperfect so as not to introduce problems with incorrectly learning how moves should be performed, but this would at least allow more information to collected as to the user's movement ability. If this concept of waypoint stages were to be implemented, it may be that a combination of state machine and waypoint detection is the better way to handle ongoing gesture detection - state machines to handle the overall movement detection, with waypoints used to provide an intuitive way to detect specific movements. The substages of the gesture could further be linked to the concept of contextual feedback for a gesture attempt, briefly noted in Section 5.16.6.2 - the state machine may have 'failure' sub-states that have textual or graphical feedback attached to them for assisting the player.

6.5.3 3D Gesture Animation

There is also the potential for work to be undertaken in regards to the on-screen visualisation of the gestures within the game environment. While simple gestures can be explained with a simple on-screen phrase to describe what the player should do, some more complex or abstract actions may require a demonstration from a therapist to fully understand. As the system stands currently, to replicate this assistance in-game would require a pre-recorded video or piece of artwork to be created that can be shown on screen at the appropriate time. As this would have to be done for every new move, this would be time consuming to achieve; it would be better if there was a way of automatically generating the animations of an on-screen 3D character, as this would allow for an unlimited number of moves to be displayed on screen.

Modern gaming software often utilises a form of *inverse kinematics* to control the animations of the characters [Zon] [Epi]- a system whereby artist created anima-

tions are augmented with real time algorithms to modify them in response to in-game changes; for example a characters walking animations being automatically adjusted to match the angle of the surface they are walking on. Character inverse kinematics is generally performed by adjusting the joints in a characters mesh until some constraint is met - the foot lying at the correct angle in the character walking example. By performing inverse kinematics with the constraint of the hand being in the orientation recorded in the gesture, it would potentially be possible to animate a character to show the movement to be completed. Research will have to be undertaken to see the exact form of inverse kinematics required for the gestures, and whether any additional data will be required to be recorded in the gesture movement files for the resulting animations to be effective.

6.5.4 Therapy Personalisation

Rehabilitation from any ailment is a long process, and ensuring that the patient's needs are met throughout their time using a software delivered therapy is of great importance. As a patient makes progress through their rehabilitation, these needs may change, so any software delivered therapy should be capable of dynamically adjusting to any such changes. A rehabilitation game should also provide a sense of reward when playing it so as to encourage engagement with the therapy; this means that the game should neither be too hard nor too easy, as frustration and boredom can lead to lower engagement over time.

Computer delivered therapy has the potential benefit of being able to record every interaction made with it, affording a permanent record of a player's performance over time. This data can then be analysed for trends and improvements, and provide an insight into the player's needs. For instance, recording average gameplay session time, and which stage of the game they stopped playing at (whether this be a particular 'level' of the game, or a specific movement to perform etc) allows for certain trends to be determined - if a player is shown to stop playing at a certain point many times, then this is a sign that some factor is causing the player to decide to cease playing. Likewise, if a player's average score is constantly getting better coupled with shorter gameplay sessions, then this would perhaps indicate that the game is too easy, and that the player is becoming bored. Analysis of player experience over time is becoming a popular tool employed by professional game developers, having been used to tune the user experience of games such as Ubisoft's *Assassin's Creed* [Dan14], and is becoming a research topic in its own right [GEnS].

In addition to adjustments to the gameplay experience to keep the player motivated to play the game, data analytics can be used to shape the overall direction of the therapy, within the constraints of the therapists desired outcome. This customisa-

tion should ideally be an active component of the gameplay from the start; perhaps via an introductory stage of gameplay that is used to determine baseline levels of ability in the skills required for success within the game. It is common for video games to have a 'tutorial' level where gameplay mechanics are introduced, often with the player unable to continue to the game proper until they have shown some aptitude for the gameplay mechanics, such as aiming or jumping. By taking this concept from traditional video games, a rehabilitative game would have the ability to provide some test scenarios relating to their physical ability, the results of which can be used to personalise the ongoing gameplay experience - starting them off with simpler or slower motions to complete to succeed if necessary, and so on. While traditional video games will present the tutorial level to the player only upon starting a new play though of the game entirely, a rehabilitative-focussed game could instead ask the user to 'warm up' upon starting the game, presenting the user with a consistent environment in which to gauge their ability to perform the tasks required of them, with this data being used to make changes in the tasks presented to the user in the session, and also to flag significant degradations or improvement in ability, perhaps via email or some web-based portal, so that a patients therapists are aware of their progress at all times.

Another common video game trope is to keep track of many in-game metrics, and providing on-screen feedback when any of them have been improved upon - whether this be an improved personal best at a lap time in a racing game, to a record number of enemies eliminated within a preset time limit. Often this is matched up with in-game unlockable content, such as badges to be displayed on the user's in-game profile, or new customisable items for their game character to wear or use. Such a system would easily be portable into therapy games; as they are generally based around a repetitive set of motions testing 'real life' ability, it can easily be seen that providing feedback to the user that they have lifted their paretic arm higher than ever before, or that they have performed a certain gesture faster than ever could be tracked within the game software, and tied to on-screen events and personalised feedback. This would require maintaining a per-user database, either on the user's machine, or perhaps tied in to an online system - perhaps the same portal that would allow therapists to see their patient's performance.

As part of the data collected during the Asteroid Game trials described in this thesis, all user input, and the effect it had upon the state of the game was collected. This included user's input, how long each target took to hit and track, the scores generated, and how long the trials took to complete. By performing analysis of this data, it may be possible to determine trends in the gameplay, such as at which point (if any) in-game performance starts to suffer from user fatigue, and also whether

variations in the in-game tasks asked of them have any significance; for example, whether being asked to aim for a target on the left of the screen twice in a row causes the user to predict that a third left-side target will appear, and whether the user's performance is negatively impacted if this is not the case. If a user's ability is to be measured via in-game performance, even such small deviations should be accounted for, to ensure that the software provides an accurate portrayal of the user's current condition.

To take further advantage of the benefits that data analytics could have on patient rehabilitation, research should be undertaken over what data should be stored, and how best to process it. Future work to undertake on the software described in this thesis will centre around the creation of a method by which rehabilitation games can communicate with an external website, using web technologies to allow user performance data to be stored and analysed, both for the purposes of monitoring patient therapeutic progress, but also to provide a constant storage mechanism so that progress can be stored, and ongoing metrics fed back into the game, allowing personal improvements in such metrics to be presented to the player in the form of achievements and unlockable items. How best to improve the software to best facilitate this move will require more research into the best parameters to store, and how best to open this data up and present to therapists for their own monitoring of progress, but it can be seen how such a service would provide an ongoing service to those suffering from physical impairments.

Bibliography

[AEL+00] Tom Artursson, Tomas Eklv, Ingemar Lundstrm, Per Mrtensson, Michael Sjstrm, and Martin Holmberg. Drift correction for gas sensors using multivariate methods. *Journal of Chemometrics*, 14(5-6):711–723, sep 2000.

[ALMK10a] Gazihan Alankus, Amanda Lazar, Matt May, and Caitlin Kelleher. Towards customizable games for stroke rehabilitation. *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, (2113):2113, 2010.

[ALMK10b] Gazihan Alankus, Amanda Lazar, Matt May, and Caitlin Kelleher. Towards customizable games for stroke rehabilitation. *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, (2113):2113, 2010.

[AMB+04] S.V. Adamovich, A.S. Merians, R Boian, M Tremaine, G.S. Burdea, M Recce, and H Poizner. A virtual reality based exercise system for hand rehabilitation post-stroke: transfer to function. In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 4, pages 4936–4939. IEEE, 2004.

[Ana06] Analog Devices. Adxl 330. *Technology*, pages 1–16, 2006.

[And98] R. E. Andersen. Relationship of Physical Activity and Television Watching With Body Weight and Level of Fatness Among Children: Results From the Third National Health and Nutrition Examination Survey. *JAMA: The Journal of the American Medical Association*, 279(12):938–942, 1998.

[Ann10] Leonard a. Annetta. The I's have it: A framework for serious educational game design. *Review of General Psychology*, 14(2):105–112, 2010.

[APP+11] Maayan Agmon, Cynthia K. Perry, Elizabeth Phelan, George Demiris, and Huong Q. Nguyen. A Pilot Study of Wii Fit Ex-

ergames to Improve Balance in Older Adults. *Journal of Geriatric Physical Therapy*, page 1, 2011.

[APRT04] Carlyne Arnould, Massimo Penta, Anne Renders, and Jean-Louis Thonnard. ABILHAND-Kids: a measure of manual ability in children with cerebral palsy. *Neurology*, 63(6):1045–1052, 2004.

[AZS05] K George MM Alberti, Paul Zimmet, and Jonathan Shaw. The metabolic syndromea new worldwide definition. *The Lancet*, 366(9491):1059–1062, sep 2005.

[BBC+08] Kirk a. Brumels, Troy Blasius, Tyler Cortright, Daniel Oumedian, and Brent Solberg. Comparison of efficacy between traditional and video game based balance programs. *Clinical Kinesiology*, 62(4):26–31, 2008.

[BBD+03] Scott Bennie, Kathryn Bruner, Allan Dizon, Holly Fritz, Bob Goodman, and Saundra Peterson. Measurements of Balance: Comparison of the Timed "Up and Go" Test and Functional Reach Test with the Berg Balance Scale. *Journal of Physical Therapy Science*, 15(2):93–97, 2003.

[BBG99] S Brauer, Y Burns, and P Galley. Lateral reach: a clinical measure of medio-lateral postural stability. *Physiotherapy research international : the journal for researchers and clinicians in physical therapy*, 4(2):81–88, 1999.

[BBPB02] Mourad Bouzit, Grigore Burdea, George Popescu, and Rares Boian. The Rutgers Master II - New design force-feedback glove. *IEEE ASME Transactions on Mechatronics*, 7(2):256–263, 2002.

[BCM+08] Grigore C. Burdea, Daniel Cioi, Joseph Martin, Devin Fensterheim, and Maeve Iolenski. The Rutgers Arm II Rehabilitation System. In *ti.rutgers.edu*, volume 9, pages 4244–4244, 2008.

[BDN+07] Aimee L Betker, Ankur Desai, Cristabel Nett, Naaz Kapadia, and Tony Szturm. Game-based exercises for dynamic short-sitting balance rehabilitation of people with chronic spinal cord and traumatic brain injuries. *Physical therapy*, 87(10):1389–1398, 2007.

[BDRG10] Ryan S.J.d. Baker, Sidney K. D'Mello, Ma.Mercedes T. Rodrigo, and Arthur C. Graesser. Better to be frustrated than bored:

The incidence, persistence, and impact of learners' cognitiveaffective states during interactions with three different computer-based learning environments. *International Journal of Human-Computer Studies*, 68(4):223–241, apr 2010.

[Ber67] Nikolai Bernstein. *Coordination and regulation of movement*. Permagon Press, 1967.

[BGS+04] Susan Barreca, Carolyn (Kelly) Gowland, Paul Stratford, Maria Huijbregts, Jeremy Griffiths, Wendy Torresin, Magen Dunkley, Patricia Miller, and Lisa Masters. Development of the Chedoke Arm and Hand Activity Inventory: Theoretical Constructs, Item Generation, and Selection. *Topics in Stroke Rehabilitation*, 11(4):31–42, 2004.

[BM01] A Bardorfer and Marko Munih. Upper limb motion analysis using haptic interface. *Mechatronics, IEEE/ASME Transactions on*, 6(3):253 – 260, 2001.

[BMC+09] J. W. Burke, M. D J McNeill, D. K. Charles, P. J. Morrow, J. H. Crosbie, and S. M. McDonough. Optimising engagement for stroke rehabilitation using serious games. *Visual Computer*, 25(12):1085–1099, 2009.

[Bon07] Julio Bonis. Acute Wiiitis. *The New England journal of medicine*, 356(23):2431–2432, 2007.

[BSL+05] Susan R. Barreca, Paul W. Stratford, Cynthia L. Lambert, Lisa M. Masters, and David L. Streiner. Test-Retest reliability, validity, and sensitivity of the chedoke arm and hand activity inventory: A new measure of upper-limb function for survivors of stroke. *Archives of Physical Medicine and Rehabilitation*, 86(8):1616–1622, 2005.

[BSM+] Susan Barreca, Paul Stratford, Lisa Masters, Carolyn Gowland, Cynthia Lambert, Jeremy Griffiths, Cathy McBay, Magen Dunkley, Pat Miller, Maria Huibregts, and Wendy Torresin. *The Chedoke Arm and Hand Activity Inventory: Administration Guidelines*.

[BTB14] Harrison L Bartlett, Lena H Ting, and Jeffrey T Bingham. Accuracy of force and center of pressure measures of the Wii Balance Board. *Gait & Posture*, 39(1):224–228, jan 2014.

[BWDWG89] K. Berg, S. Wood-Dauphinee, J. I. Williams, and D. Gayton. Measuring balance in the elderly: Preliminary development of an instrument, 1989.

[CAL14] Hsiu-Ching Chiu, Louise Ada, and Hsien-Min Lee. Upper limb training using Wii Sports Resort for children with hemiplegic cerebral palsy: A randomized, single-blind trial. *Clinical rehabilitation*, may 2014.

[CBP+10] Ross a Clark, Adam L Bryant, Yonghao Pua, Paul McCrory, Kim Bennell, and Michael Hunt. Validity and reliability of the Nintendo Wii Balance Board for assessment of standing balance. *Gait & posture*, 31(3):307–10, mar 2010.

[CCH11] Yao-Jen Chang, Shu-Fang Chen, and Jun-Da Huang. A Kinect-based system for physical rehabilitation: a pilot study for young adults with motor disabilities. *Research in developmental disabilities*, 32(6):2566–70, 2011.

[Che07] Jenova Chen. Flow in games (and everything else). *Communications of the ACM*, 50(4):31, 2007.

[CLBM07] J H Crosbie, S Lennon, J R Basford, and S M McDonough. Virtual reality in stroke rehabilitation: still more virtual than real. *Disability and rehabilitation*, 29(14):1139–1146; discussion 1147–1152, 2007.

[CLZ12] Cy Chang, Belinda Lange, and M Zhang. Towards pervasive physical rehabilitation using Microsoft Kinect. *Pervasive Health*, pages 159–162, 2012.

[CM07] V.S. Chouhan and S.S. Mehta. Total Removal of Baseline Drift from ECG Signal. In *2007 International Conference on Computing: Theory and Applications (ICCTA'07)*, pages 512–515. IEEE, mar 2007.

[CTR+15] David J Clarke, Sarah Tyson, Helen Rodgers, Avril Drummond, Rebecca Palmer, Matthew Prescott, Pippa Tyrrell, Louisa Burton, Katie Grenfell, Lianne Brkic, and Anne Forster. Why do patients with stroke not receive the recommended amount of active therapy (ReAcT)? Study protocol for a multisite case study investigation. *BMJ open*, 5(8):e008443, 2015.

[Dan14] Jonathan Dankoff. Game Telemetry with DNA Tracking on Assassin's Creed, 2014.

[DBF+08] Judith E Deutsch, Megan Borbely, Jenny Filler, Karen Huhn, and Phyllis Guarrera-Bowlby. Use of a low-cost, commercially available gaming console (Wii) for rehabilitation of an adolescent with cerebral palsy. *Physical therapy*, 88(10):1196–207, oct 2008.

[DBS+11] Judith E Deutsch, Arielle Brettler, Caroline Smith, Jamie Welsh, Roshan John, Phyllis Guarrera-Bowlby, and Michal Kafri. Nintendo wii sports and wii fit game analysis, validation, and application to stroke rehabilitation. *Topics in stroke rehabilitation*, 18(6):701–19, 2011.

[Dev] Analog Devices. ADXL330 Accelerometer Datasheet.

[DG85] W H Dietz and S L Gortmaker. Do we fatten our children at the television set? Obesity and television viewing in children and adolescents. *Pediatrics*, 75(5):807–812, 1985.

[DHHW13] Patrick S. Dukes, Austen Hayes, Larry F. Hodges, and Michelle Woodbury. Punching ducks for post-stroke neurorehabilitation: System design and initial exploratory feasibility study. *IEEE Symposium on 3D User Interface 2013, 3DUI 2013 - Proceedings*, pages 47–54, 2013.

[DLLP09] Jilyan Decker, Harmony Li, Dan Losowyj, and Vivek Prakash. Wiihabilitation : Rehabilitation of Wrist Flexion and Extension Using a Wiimote-Based Game System. *Governor's School of Engineering and Technology Research Journal*, pages 92–98, 2009.

[DLP] DLP Design. DLP IO8.

[DPN83] P W Duncan, M Propst, and S G Nelson. Reliability of the Fugl-Meyer assessment of sensorimotor recovery following cerebrovascular accident. *Physical therapy*, 63(10):1606–10, oct 1983.

[DRW+98] Pamela Duncan, Lorie Richards, Dennis Wallace, J. Stoker-Yates, Patricia Pohl, Carl Luchies, Abna Ogle, and Stephanie Studenski. A Randomized, Controlled Pilot Study of a Home-Based Exercise Program for Individuals With Mild and Moderate Stroke. *Stroke*, 29(10):2055–2060, oct 1998.

[DS97] Catherine M Dean and Roberta B Shepherd. Task-Related Training Improves Performance of Seated Reaching Tasks After

Stroke: A Randomized Controlled Trial. *Stroke*, 28(4):722–728, apr 1997.

[DSB+11] Pamela W Duncan, Katherine J Sullivan, Andrea L Behrman, Stanley P Azen, Samuel S Wu, Stephen E Nadeau, Bruce H Dobkin, Dorian K Rose, Julie K Tilson, Steven Cen, and Sarah K Hayden. Body-weight-supported treadmill rehabilitation after stroke. *The New England journal of medicine*, 364(21):2026–2036, 2011.

[DSR+03] Pamela Duncan, Stephanie Studenski, Lorie Richards, Steven Gollub, Sue Min Lai, Dean Reker, Subashan Perera, Joni Yates, Victoria Koch, Sally Rigler, and Dallas Johnson. Randomized clinical trial of therapeutic exercise in subacute stroke. *Stroke; a journal of cerebral circulation*, 34(9):2173–80, sep 2003.

[DSTJ13] Colin A Depp, David A Schkade, Wesley K Thomspon, and Dilip V Jeste. Age, Affective Experience, and Television Use. *American journal of Preventive Medicine*, 39(2):173–178, 2013.

[DWCS90] P W Duncan, D K Weiner, J Chandler, and S Studenski. Functional reach: a new clinical measure of balance. *Journal of gerontology*, 45(6):M192–M197, 1990.

[DYC+10] Dangxiao Wang, Yuru Zhang, Chong Yao, Jun Wu, Huimin Jiao, and Muli Liu. Toward Force-Based Signature Verification: A Pen-Type Sensor and Preliminary Validation. *IEEE Transactions on Instrumentation and Measurement*, 59(4):752–762, apr 2010.

[EC13] National Institute for Health Excellence and Care. Stroke Rehabilitation: Long-term rehabilitation after stroke. Technical report, National Institute for Health and Care Excellence, 2013.

[ECE+11] Gammon M Earhart, Jim T Cavanaugh, Terry Ellis, Matt P Ford, K Bo Foreman, and Lee Dibble. The 9-hole PEG test of upper extremity function: average values, test-retest reliability, and factors contributing to performance in people with Parkinson disease. *Journal of neurologic physical therapy : JNPT*, 35(4):157–63, 2011.

[Ent14] Entertainment Software Association. 2014 Essential Facts. Technical report, Entertainment Software Association, 2014.

[Epi] Inc. Epic Games. Unreal Engine - IK Setups - https://docs.unrealengine.com/latest/INT/Engine/Animation/IKSetups/.

[FBSL12] Adso Fernández-Baena, Antonio Susín, and Xavier Lligadas. Biomechanical validation of upper-body and lower-body joint movements of kinect motion capture data for rehabilitation treatments. *Proceedings of the 2012 4th International Conference on Intelligent Networking and Collaborative Systems, INCoS 2012*, pages 656–661, 2012.

[FHE+05] Robert S. Fisher, Graham Harding, Giuseppe Erba, Gregory L. Barkley, and Arnold Wilkins. Photic- and Pattern-induced Seizures: A Review for the Epilepsy Foundation of America Working Group. *Epilepsia*, 46(9):1426–1441, sep 2005.

[FHS+12] Vera Fung, Aileen Ho, Jennifer Shaffer, Esther Chung, and Manuel Gomez. Use of Nintendo Wii Fit In the rehabilitation of outpatients following total knee replacement: A preliminary randomised controlled trial. *Physiotherapy (United Kingdom)*, 98(3):183–188, 2012.

[FLG11] M. Fluet, Olivier Lambercy, and Roger Gassert. Upper limb assessment using a Virtual Peg Insertion Test. In *2011 IEEE International Conference on Rehabilitation Robotics*, pages 1–6. IEEE, jun 2011.

[FMJL+75] A R Fugl-Meyer, L Jääskö, I Leyman, S Olsson, and S Steglind. The post-stroke hemiplegic patient. 1. a method for evaluation of physical performance. *Scandinavian journal of rehabilitation medicine*, 7(1):13–31, 1975.

[Foo13] Food and Drug Administration/ U.S. Department of Health and Human. Guidance for Industry and Food. Technical report, 2013.

[FPB07] Sheryl Flynn, Phyllis Palma, and Anneke Bender. Feasibility of using the Sony PlayStation 2 gaming platform for an individual poststroke: a case report. *Journal of neurologic physical therapy : JNPT*, 31(4):180–9, dec 2007.

[Gam] GameSpot. Dance Dance Revolution hits 6.5 million in sales - http://www.gamespot.com/articles/dance-dance-revolution-hits-65-million-in-sales/1100-6084894/.

[Gar13] Gartner. Forecast: Video Game Ecosystem, Worldwide, 4Q13. Technical report, 2013.

[GDB02] David J Gladstone, Cynthia J Danells, and Sandra E Black. The fugl-meyer assessment of motor recovery after stroke: a critical review of its measurement properties. *Neurorehabilitation and neural repair*, 16(3):232–240, 2002.

[GEnS] André R Gagné, Magy Seif El-nasr, and Chris D Shaw. Gagné, El-nasr, Shaw - 2011 - A Deeper Look at the Use of Telemetry for Analysis of Player Behavior in RTS.

[GGLAC11] José-Antonio Gil-Gómez, Roberto Lloréns, Mariano Alcañiz, and Carolina Colomer. Effectiveness of a Wii balance board-based system (eBaViR) for balance rehabilitation: a pilot randomized clinical trial in patients with acquired brain injury. *Journal of neuroengineering and rehabilitation*, 8(1):30, jan 2011.

[GMR+13] Alan S Go, Dariush Mozaffarian, Véronique L Roger, Emelia J Benjamin, Jarett D Berry, William B Borden, Dawn M Bravata, Shifan Dai, Earl S Ford, Caroline S Fox, Sheila Franco, Heather J Fullerton, Cathleen Gillespie, Susan M Hailpern, John a Heit, Virginia J Howard, Mark D Huffman, Brett M Kissela, Steven J Kittner, Daniel T Lackland, Judith H Lichtman, Lynda D Lisabeth, David Magid, Gregory M Marcus, Ariane Marelli, David B Matchar, Darren K McGuire, Emile R Mohler, Claudia S Moy, Michael E Mussolino, Graham Nichol, Nina P Paynter, Pamela J Schreiner, Paul D Sorlie, Joel Stein, Tanya N Turan, Salim S Virani, Nathan D Wong, Daniel Woo, and Melanie B Turner. Heart disease and stroke statistics–2013 update: a report from the American Heart Association. *Circulation*, 127(1):e6–e245, jan 2013.

[GMS+08] Kaushal S Gandhi, Fiona C McKay, Stephen D Schibeci, Jonathan W Arthur, Rob N Heard, Graeme J Stewart, and David R Booth. BAFF is a biological response marker to IFN-beta treatment in multiple sclerosis. *Journal of interferon & cytokine research : the official journal of the International Society for Interferon and Cytokine Research*, 28(9):529–539, 2008.

[GMW+10] Meredith R Golomb, Brenna C McDonald, Stuart J Warden, Janell Yonkman, Andrew J Saykin, Bridget Shirley, Meghan

Huber, Bryan Rabin, Moustafa Abdelbaky, Michelle E Nwosu, Monica Barkat-Masih, and Grigore C Burdea. In-home virtual reality videogame telerehabilitation in adolescents with hemiplegic cerebral palsy. *Archives of physical medicine and rehabilitation*, 91(1):1–8.e1, jan 2010.

[GODPMZAR14] D. González-Ortega, F. J. Díaz-Pernas, M. Martínez-Zarzuela, and M. Antón-Rodríguez. A Kinect-based system for cognitive rehabilitation exercises monitoring. *Computer Methods and Programs in Biomedicine*, 113(2):620–631, 2014.

[GPHS09] Diana L Graf, Lauren V Pratt, Casey N Hester, and Kevin R Short. Playing active video games increases energy expenditure in children. *Pediatrics*, 124:534–540, 2009.

[Gro10] Information Solutions Group. 2010 Social Gaming Research Information Solutions Group. 2010.

[GS97] L B Goldstein and G P Samsa. Reliability of the National Institutes of Health Stroke Scale. Extension to non-neurologists in the context of a clinical trial. *Stroke; a journal of cerebral circulation*, 28(2):307–10, feb 1997.

[GTD10] Louise a Gustafsson, Merrill J Turpin, and Caitlin M Dorman. Clinical utility of the Chedoke Arm and Hand Activity Inventory for stroke rehabilitation. *Canadian journal of occupational therapy. Revue canadienne d'ergotherapie*, 77(3):167–173, 2010.

[Han87] P K Hansen. Method and apparatus for position and orientation measurement using a magnetic field and retransmission, 1987.

[HCDC04] Ramona Hicks, Paige Cook, Tracy Dulas, and Janelle Clem. Demographics of physical therapy practice: implications for education. *Journal of Physical Therapy Education*, 18(2):80–86, 2004.

[HLD12] Margaret Jean Hall, Shaleah Levant, and Carol J DeFrances. Hospitalization for stroke in U.S. hospitals, 1989-2009. *NCHS data brief*, (95):1–8, 2012.

[HLY+14] Ming Chun Huang, Si Huei Lee, Shih Ching Yeh, Rai Chi Chan, Albert Rizzo, Wenyao Xu, Han Lin Wu, and Shan Hui Lin. Intelligent frozen shoulder rehabilitation. *IEEE Intelligent Systems*, 29(3):22–28, 2014.

[HMHD09] Alexandra Handley, Pippa Medcalf, Kate Hellier, and Dipankar Dutta. Movement disorders after stroke. *Age and Ageing*, 38(3):260–266, 2009.

[HMS+13] Nathan B. Herz, Shyamal H. Mehta, Kapil D. Sethi, Paula Jackson, Patricia Hall, and John C. Morgan. Nintendo Wii rehabilitation (Wii-hab) provides benefits in Parkinson's disease. *Parkinsonism & Related Disorders*, 19(11):1039–1042, nov 2013.

[Hos] Oxford University Hospitals. Helping your wrist to recover after a fracture. Technical report, Oxford University Hospitals NHS Trust.

[HPBM01] M L Harris, M I Polkey, P M Bath, and J Moxham. Quadriceps muscle weakness following acute hemiplegic stroke. *Clinical rehabilitation*, 15(01):274–281, 2001.

[HVL10] Michael Hoffman, Paul Varcholik, and Joseph J. LaViola. Breaking the status quo: Improving 3D gesture recognition with spatially convenient input devices. *Proceedings - IEEE Virtual Reality*, pages 59–66, 2010.

[HWW+87] Andrew Heller, Derick T. Wade, Victorine A. Wood, Alan Sunderland, R. L. Hewer, and Elizabeth Ward. Arm function after stroke: measurement and recovery over the first three months. *Journal of neurology, neurosurgery, and psychiatry*, 50(6):714–9, 1987.

[IAH11] Di Loreto Ines, Gouaich Abdelkader, and Nadia Hocine. Mixed reality serious games for post-stroke rehabilitation. *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 530–537, 2011.

[IEP99] J L Ingles, G A Eskes, and S J Phillips. Fatigue after stroke. *Archives of physical medicine and rehabilitation*, 80(2):173–8, feb 1999.

[JBM+01] David Jack, Rares Boian, Alma S. Merians, Marilyn Tremaine, Grigore C. Burdea, Sergei V. Adamovich, Michael Recce, and Howard Poizner. Virtual reality-enhanced stroke rehabilitation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 9(3):308–318, 2001.

[JMMG10] Kirsten Jack, Sionnadh Mairi McLean, Jennifer Klaber Moffett, and Eric Gardiner. Barriers to treatment adherence in physiotherapy outpatient clinics: A systematic review. *Manual Therapy*, 15(3):220–228, 2010.

[JPFJS13] Jennifer Jelsma, Marieke Pronk, Gillian Ferguson, and Dorothee Jelsma-Smit. The effect of the Nintendo Wii Fit on balance control and gross motor function of children with spastic hemiplegic cerebral palsy. *Developmental neurorehabilitation*, 16(1):27–37, jan 2013.

[JTM+13] Ard Jacobs, Annick Timmermans, Marc Michielsen, Maaiken Vander Plaetse, Panos Markopoulos, Maaiken Vander Plaetse, Rehabilitation Medicine, Jessa Hospital, and Campus Sint. CONTRAST: gamification of arm-hand training for stroke survivors. *CHI EA '13: CHI '13 Extended Abstracts on Human Factors in Computing Systems*, page 415, 2013.

[JvdWN+08] Michiel J a Jannink, Gelske J van der Wilden, Dorine W Navis, Gerben Visser, Jeanine Gussinklo, and Maarten Ijzerman. A low-cost video game applied for training of upper extremity function in children with cerebral palsy: a pilot study. *Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, 11(1):27–32, feb 2008.

[KBM+06] Manjuladevi Kuttuva, Rares Boian, Alma Merians, Grigore Burdea, Mourad Bouzit, Jeffrey Lewis, and Devin Fensterheim. The Rutgers Arm, a Rehabilitation System in Virtual Reality: A Pilot Study. *CyberPsychology & Behavior*, 9(2):148–152, apr 2006.

[KE12] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[KFCR06] Derek G. Kamper, Heidi C. Fischer, Erik G. Cruz, and William Z. Rymer. Weakness Is the Primary Contributor to Finger Impairment in Chronic Stroke. *Archives of Physical Medicine and Rehabilitation*, 87(9):1262–1269, sep 2006.

[KFK+13] Anne D Kloos, Nora E Fritz, Sandra K Kostyk, Gregory S Young, and Deb a Kegelmeyer. Video game play (Dance Dance Revolution) as a potential exercise therapy in Huntington's disease: a controlled clinical trial. *Clinical rehabilitation*, 27(11):972–82, nov 2013.

[KHJ+14] Bo Ryun Kim, Eun Young Han, Seung Jae Joo, Song Yi Kim, and Ho Min Yoon. Cardiovascular fitness as a predictor of functional recovery in subacute stroke patients. *Disability and rehabilitation*, 36(3):227–31, 2014.

[kin] Kinect hardware - https://developer.microsoft.com/en-us/windows/kinect/hardware.

[KJ08] Jeffrey a Kleim and Theresa a Jones. Principles of experience-dependent neural plasticity: implications for rehabilitation after brain damage. *Journal of speech, language, and hearing research : JSLHR*, 51(1):S225–39, 2008.

[KJ11] Stuart T Klapp and Richard J Jagacinski. Gestalt principles in the control of motor action. *Psychological bulletin*, 137(3):443–462, 2011.

[Kla03] Stuart T Klapp. Reaction time analysis of two types of motor preparation for speech articulation: action as a sequence of chunks. *Journal of motor behavior*, 35(March 2015):135–150, 2003.

[KSL07] Louis Kratz, Matthew Smith, and Frank J Lee. Wiizards: 3D gesture recognition for game play input. *Proceedings of the 2007 conference on Future Play Future Play 07*, pages 209–212, 2007.

[Kvi04] Joanna Kvist. Rehabilitation following anterior cruciate ligament injury: current recommendations for sports participation. *Sports medicine (Auckland, N.Z.)*, 34(4):269–280, 2004.

[KWK+97] G. Kwakkel, R. C. Wagenaar, T. W. Koelman, G. J. Lankhorst, and J. C. Koetsier. Effects of intensity of rehabilitation after stroke. A research synthesis. *Stroke; a journal of cerebral circulation*, 28(8):1550–6, aug 1997.

[Las03] Bryan Lask. Motivating children and adolescents to improve adherence. *The Journal of Pediatrics*, 143(4):430–433, oct 2003.

[LCS+11] Belinda Lange, Chien-Yen Chang, Evan Suma, Bradley Newman, Albert Skip Rizzo, and Mark Bolas. Development and evaluation of low cost game-based balance rehabilitation tool using the Microsoft Kinect sensor. *Conference proceedings : Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine*

and Biology Society. Annual Conference, 2011:1831–4, aug 2011.

[LFJF+06] Lorraine Lanningham-Foster, Teresa B Jensen, Randal C Foster, Aoife B Redmond, Brian a Walker, Dieter Heinz, and James a Levine. Energy expenditure of sedentary screen time compared with active screen time for children. *Pediatrics*, 118(6):e1831–e1835, 2006.

[LFR09] B. Lange, S. Flynn, and a. Rizzo. Initial usability assessment of off-the-shelf video game consoles for clinical game-based motor rehabilitation. *Physical Therapy Reviews*, 14(5):355–363, oct 2009.

[LJ11] Max A Little and Nick S Jones. Generalized methods and solvers for noise removal from piecewise constant signals. I. Background theory. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 467(2135):3088–3114, sep 2011.

[LZWV09] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[MBK+08] Ann E Maloney, T Carter Bethea, Kristine S Kelsey, Julie T Marks, Sadye Paez, Angela M Rosenberg, Diane J Catellier, Robert M Hamer, and Linmarie Sikich. A pilot of a video game (DDR) to promote physical activity and decrease sedentary screen time. *Obesity (Silver Spring, Md.)*, 16(9):2074–2080, 2008.

[MCN+09] Emily C-S Murphy, Linda Carson, William Neal, Christine Baylis, David Donley, and Rachel Yeater. Effects of an exercise intervention using Dance Dance Revolution on endothelial function and other risk factors in overweight children. *International journal of pediatric obesity : IJPO : an official journal of the International Association for the Study of Obesity*, 4(September 2008):205–214, 2009.

[MDBM06] Kira Morrow, Ciprian Docan, Grigore Burdea, and Alma Merians. Low-cost Virtual Rehabilitation of the Hand for Patients Post-Stroke. *2006 International Workshop on Virtual Rehabilitation*, pages 6–10, 2006.

[MDMM11] Marie R Mouawad, Catherine G Doust, Madeleine D Max, and Penelope a McNulty. Wii-based movement therapy to promote improved upper extremity function post-stroke: a pilot study. *Journal of rehabilitation medicine*, 43(6):527–33, may 2011.

[MJB+02] Alma S. Merians, David Jack, Rares Boian, Marilyn Tremaine, Grigore C Burdea, Sergei V Adamovich, Michael Recce, and Howard Poizner. Virtual reality-augmented rehabilitation for patients following stroke. *Physical therapy*, 82(9):898–915, sep 2002.

[MJM05] W. Morley, K. Jackson, and Gillian E. Mead. Post-stroke fatigue: An important yet neglected symptom. *Age and Ageing*, 34(3):313, 2005.

[MMJ+07] Ralph Maddison, Cliona Ni Mhurchu, Andrew Jull, Yannan Jiang, Harry Prapavessis, and Anthony Rodgers. Energy expended playing video console games: an opportunity to increase children's physical activity? *Pediatric exercise science*, 19(3):334–343, 2007.

[MWKV85] Virgil Mathiowetz, Karen Weber, Nancy Kashman, and Gloria Volland. Adult norms for the Nine Hole Peg Test of finger dexterity. *Occupational Therapy Journal of Research*, 5(1):24–38, 1985.

[MWRA+04] Nicola Massy-Westropp, Wayne Rankin, Michael Ahern, Jegan Krishnan, and Trevor C. Hearn. Measuring grip strength in normal adults: Reference ranges and a comparison of electronic and hydraulic instruments. *The Journal of Hand Surgery*, 29(3):514–519, may 2004.

[Nag89] H. Nagasaki. Asymmetric velocity and acceleration profiles of human arm movements. *Experimental Brain Research*, 74(2):319–326, 1989.

[nhsa] NHS Choices - Physiotherapy.

[NHSb] NHS. NHS Choices - Physiotherapy Techniques.

[NHSc] NHS. Stroke - Symptoms.

[nih13] Hall Effect Accelerometer. Technical report, National Instruments, 2013.

[Nin] Nintendo. It's official! Nintendo's Wii Balance Board is a record breaker! https://www.nintendo.co.uk/News/2012/It-s-official-Nintendo-s-Wii-Balance-Board-is-a-record-breaker–253133.html.

[NS03] Kai Nickel and Rainer Stiefelhagen. Pointing Gesture Recognition based on 3D-Tracking of Face , Hands and Head Orientation Categories and Subject Descriptors. *Proceedings of the 5th international conference on Multimodal interfaces*, pages 140–146, 2003.

[OEM06] Alison J Orrell, Frank F Eves, and Rich S W Masters. Motor learning of a dynamic balancing task after stroke: implicit implications for stroke rehabilitation. *Physical therapy*, 86(3):369–80, mar 2006.

[ORH14] C. O'Donovan, E. F. Roche, and J. Hussey. The energy cost of playing active video games in children with obesity and children of a healthy weight. *Pediatric Obesity*, 9(4):310–317, 2014.

[PBT+05] Janet L. Poole, Patricia a. Burtner, Theresa a. Torres, Cheryl Kirk McMullen, Amy Markham, Michelle Lee Marcum, Jennifer Bradley Anderson, and Clifford Qualls. Measuring dexterity in children using the Nine-hole Peg Test. *Journal of Hand Therapy*, 18(3):348–351, 2005.

[PPvW+05] Thomas Platz, Cosima Pinkowski, Frederike van Wijck, In-Ha Kim, Paolo di Bella, and Garth Johnson. Reliability and validity of arm function assessment with standardized guidelines for the Fugl-Meyer Test, Action Research Arm Test and Box and Block Test: a multicentre study. *Clinical rehabilitation*, 19(4):404–411, 2005.

[PR91] D Podsiadlo and S Richardson. The timed "Up & Go": a test of basic functional mobility for frail elderly persons. *Journal of the American Geriatrics Society*, 39(2):142–148, 1991.

[PRR10] Andrew K. Przybylski, C. Scott Rigby, and Richard M. Ryan. A motivational model of video game engagement. *Review of General Psychology*, 14(2):154, 2010.

[PTA+01] M Penta, L Tesio, C Arnould, A Zancan, and J.-L. Thonnard. The ABILHAND Questionnaire as a Measure of Manual Ability

in Chronic Stroke Patients : Rasch-Based Validation and Relationship to Upper Limb Impairment. *Stroke*, 32(7):1627–1634, jul 2001.

[PTA+10] Lamberto Piron, Andrea Turolla, Michela Agostini, Carla Silvana Zucconi, Laura Ventura, Paolo Tonin, and Mauro Dam. Motor Learning Principles for Rehabilitation: A Pilot Randomized Controlled Study in Poststroke Patients. *Neurorehabilitation and Neural Repair*, 24(6):501–508, jul 2010.

[PTT98] Massimo Penta, Jean Louis Thonnard, and Luigi Tesio. ABILHAND: A Rasch-built measure of manual ability. *Archives of Physical Medicine and Rehabilitation*, 79(9):1038–1042, 1998.

[RB95] D. Reynaerts and H. Van Brussel. Design of an advanced computer writing tool. *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 229–234, 1995.

[RCB+08] Avinash Ramchandani, Kevin Carroll, Roel Buenaventura, Jason Douglas, and Justin Liu. Wii-habilitation increases participation in therapy. *2008 Virtual Rehabilitation, IWVR*, page 69, 2008.

[RCCR01] M Randall, J B Carlin, P Chondros, and D Reddihough. Reliability of the Melbourne assessment of unilateral upper limb function. *Dev Med Child Neurol*, 43(11):761–767, 2001.

[RDPPC+12] Cristina Rodriguez-De-Pablo, Joel C. Perry, Francesca I. Cavallaro, Haritz Zabaleta, and Thierry Keller. Development of computer games for assessment and training in post-stroke arm telerehabilitation. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pages 4571–4574, 2012.

[RGT+11] Tennille Rowland, Louise Gustafsson, Merrill Turpin, Robert Henderson, and Stephen Read. Chedoke Arm and Hand Activity Inventory-9 (CAHAI-9): a multi-centre investigation of clinical utility... ...including commentary by Pooyania S and Schuster C. *International Journal of Therapy & Rehabilitation*, 18:290–298, 2011.

[RJR99] M Randall, L Johnson, and D Reddihough. The Melbourne assessment of unilateral upper limb function: test administration manual. *Melbourne: Royal Children's Hospital*, 1999.

[RLM+03] Jenny Röding, Britta Lindström, Jan Malm, Ann Öhman, and Ann Ohman. Frustrated and invisible–younger stroke patients' experiences of the rehabilitation process. *Disability and Rehabilitation*, 25(15):867–874, 2003.

[RRP06] Richard M. Ryan, C. Scott Rigby, and Andrew Przybylski. The Motivational Pull of Video Games: A Self-Determination Theory Approach. *Motivation and Emotion*, 30(4):344–360, dec 2006.

[SBM14] Christine T Shiner, Winston D Byblow, and Penelope a McNulty. Bilateral Priming Before Wii-based Movement Therapy Enhances Upper Limb Rehabilitation and Its Retention After Stroke: A Case-Controlled Study. *Neurorehabilitation and neural repair*, mar 2014.

[SBU+14] M Scott, W Blewitt, G Ushaw, J Q Shi, G Morgan, and J Eyre. Automating assessment in video game teletherapy: Data cutting. In *2014 IEEE Symposium on Computational Intelligence in Healthcare and e-health (CICARE)*, pages 9–16. IEEE, dec 2014.

[Sch07] B. J. Schaller. Influence of age on stroke and preconditioning-induced ischemic tolerance in the brain. *Experimental Neurology*, 205(1):9–19, 2007.

[SHM07] Jeff Sinclair, Philip Hingston, and Martin Masek. Considerations for the design of exergames. *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia - GRAPHITE '07*, 1(212):289–295, 2007.

[SKB07] E. Sriraghavendra, Karthik K., and Chiranjib Bhattacharyya. Frechet Distance Based Approach for Searching Online Handwritten Documents. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 1, pages 461–465. IEEE, sep 2007.

[SKSN15] Keizo Sato, Kaoru Kuroki, Syuko Saiki, and Ryoichi Nagatomi. Improving Walking, Muscle Strength, and Balance in the Elderly with an Exergame Using Kinect: A Randomized Controlled Trial. *Games for Health Journal*, 4(3):161–167, 2015.

[SLT08] Katie Sell, Tia Lillie, and Julie Taylor. Energy expenditure during physically interactive video game playing in male college

students with different playing experience. *Journal of American college health : J of ACH*, 56(5):505–511, 2008.

[SMB+10] G Saposnik, M Mamdani, M Bayley, K E Thorpe, J Hall, L G Cohen, and R Teasell. Clinical trial protocols E ffectiveness of V irtual R eality E xercises in ST roke Rehabilitation (EVREST): Rationale , Design , and Protocol of a Pilot Randomized Clinical Trial Assessing the Wii Gaming System Clinical trial protocols. 5(February):47–51, 2010.

[SME] J Serradilla J Q Shi, Y Cheng G Morgan, and J a Eyre. Statistical modelling of Upper Limb Functional Ability using Motion Data : Validity Study.

[SME14] J Serradilla J Q Shi, Y Cheng G Morgan, and J a Eyre. Automatic Assessment of Upper Limb Function During Play of the Action Video Game , Circus Challenge : Validity and Sensitivity to Change. 2014.

[SMS+93] J Sanford, J Moreland, L R Swanson, P W Stratford, and C Gowland. Reliability of the Fugl-Meyer assessment for testing motor performance in patients following stroke. *Physical therapy*, 73(7):447–454, 1993.

[SMT+03] Heidi Sveistrup, Joan McComas, Marianne Thornton, Shawn Marshall, Hillel Finestone, Anna McCormick, Kevin Babulic, and Alain Mayhew. Experimental studies of virtual reality-delivered compared to conventional exercise programs for rehabilitation. *Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, 6(3):245–9, jun 2003.

[SPH+10] S Studenski, S Perera, E Hile, V Keller, and J Garcia. Interactive Video Dance Games for Healthy Older Adults Interactive Video Dance Games for Healthy Older Adults. 14(10):850–852, 2010.

[SRM06] Marianne Shaughnessy, Barbara M Resnick, and Richard F Macko. Testing a model of post-stroke exercise behavior. *Rehabilitation nursing : the official journal of the Association of Rehabilitation Nurses*, 31(1):15–21, 2006.

[SSS+09] S T Smith, C Sherrington, S Studenski, D Schoene, and S R Lord. A novel Dance Dance Revolution (DDR) system for in-home training of stepping ability: basic parameters of system

use by older adults. *British journal of sports medicine*, 45:441–445, 2009.

[STBH89] A Sunderland, D Tinson, L Bradley, and R L Hewer. Arm function after stroke. An evaluation of grip strength as a measure of recovery and a prognostic indicator. *Journal of Neurology, Neurosurgery & Psychiatry*, 52(11):1267–1272, nov 1989.

[STC+11] Katherine J Sullivan, Julie K Tilson, Steven Y Cen, Dorian K Rose, Julie Hershberg, Anita Correa, Joann Gallichio, Molly McLeod, Craig Moore, Samuel S Wu, and Pamela W Duncan. Fugl-Meyer assessment of sensorimotor function after stroke: standardized training procedure for clinical practice and clinical trials. *Stroke; a journal of cerebral circulation*, 42(2):427–32, mar 2011.

[STM+10] Gustavo Saposnik, Robert Teasell, Muhammad Mamdani, Judith Hall, William McIlroy, Donna Cheung, Kevin E Thorpe, Leonardo G Cohen, and Mark Bayley. Effectiveness of virtual reality using Wii gaming technology in stroke rehabilitation: a pilot randomized clinical trial and proof of principle. *Stroke; a journal of cerebral circulation*, 41(7):1477–84, jul 2010.

[SVA+13] Nadja Scherbakov, Stephan Von Haehling, Stefan D. Anker, Ulrich Dirnagl, and Wolfram Doehner. Stroke induced Sarcopenia: Muscle wasting and disability after stroke. *International Journal of Cardiology*, 170(2):89–94, 2013.

[SWEBB09] Heidi Sugarman, Aviva Weisel-Eichler, Arie Burstin, and Riki Brown. Use of the Wii Fit system for the treatment of balance problems in the elderly: A feasibility study. *2009 Virtual Rehabilitation International Conference, VR 2009*, pages 111–116, 2009.

[TMC+] Edward Taub, David M Morris, Jean Crago, Danna Kay King, Mary Bowman, Camille Bryson, Staci Bishop, Sonya Pearson, and Sharon E Shaw. Wolf Motor Function Test (WMFT). Technical report.

[Tor] Linus Torvalds. index : kernel/git/torvalds/linux.git - http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=9fa4

[Tru] Gloucestershire Hospitals NHS Foundation Trust. NHS - Stroke Symptoms.

[TSONB99] Luci Fuscaldi Teixeira-Salmela, Sandra Jean Olney, Sylvie Nadeau, and Brenda Brouwer. Muscle strengthening and physical conditioning to reduce impairment and disability in chronic stroke survivors. *Archives of Physical Medicine and Rehabilitation*, 80(10):1211–1218, 1999.

[VL09] Latha Velayudhan and Simon Lovestone. Smell identification test as a treatment response marker in patients with Alzheimer disease receiving donepezil. Technical Report 4, Institute of Psychiatry, King's College London, London, United Kingdom. Latha.Velayudhan@kcl.ac.uk, 2009.

[VR16] Oculus VR. Building a Sensor for Low Latency VR, 2016.

[WBH+07] Darren E.R. Warburton, Shannon S.D. Bredin, Leslie T.L. Horita, Dominik Zbogar, Jessica M. Scott, Ben T.a. Esch, and Ryan E. Rhodes. The health benefits of interactive video game exercise. *Applied Physiology, Nutrition, and Metabolism*, 32(4):655–663, 2007.

[WCE+01] S L Wolf, P a Catlin, M Ellis, a L Archer, B Morgan, and A Piacentino. Assessing Wolf Motor Function Test as Outcome Measure for Research in Patients After Stroke. *Stroke*, 32(7):1635–1639, jul 2001.

[Wei83] S Weisman. Computer games for the frail elderly. *The Gerontologist*, 23(4):361–3, aug 1983.

[WKT+13] Diny G M Winkels, Anke I R Kottink, Rutger a J Temmink, Juliëtte M M Nijlant, and Jaap H Buurke. Wii-habilitation of upper extremity function in children with Cerebral Palsy. An explorative study. *Developmental Neurorehabilitation*, 16(1):44–51, feb 2013.

[WLBJ89] S L Wolf, D E Lecraw, L A Barton, and B B Jann. Forced use of hemiplegic upper extremities to reverse the effect of learned nonuse among chronic stroke and head-injured patients. *Experimental neurology*, 104(2):125–32, may 1989.

[YFBC11] William Young, Stuart Ferguson, Sébastien Brault, and Cathy Craig. Assessing and training standing balance in older adults: A novel approach using the 'Nintendo Wii' Balance Board. *Gait and Posture*, 33(2):303–305, 2011.

[YJK+05] Sung H You, Sung Ho Jang, Yun-Hee Kim, Yong-Hyun Kwon, Irene Barrow, and Mark Hallett. Cortical reorganization induced by virtual reality therapy in a child with hemiparetic cerebral palsy. *Developmental Medicine and Child Neurology*, 47(09):628, aug 2005.

[YSX+10] Loh Yong Joo, Tjan Soon Yin, Donald Xu, Ernest Thia, Chia Pei Fen, Christopher Wee Keong Kuah, and Keng-He Kong. A feasibility study using interactive commercial off-the-shelf computer gaming in upper limb rehabilitation in patients after stroke. *Journal of rehabilitation medicine : official journal of the UEMS European Board of Physical and Rehabilitation Medicine*, 42(5):437–41, may 2010.

[Zac97] Gabriel Zachmann. Distortion Correction of Magnetic Fields for Position Tracking. In *Computer Graphics International, 1997. Proceedings*, pages 213 – 220, 251, 1997.

[ZMW04] Christopher a Zaino, Victoria Gocha Marchese, and Sarah L Westcott. Timed up and down stairs test: preliminary reliability and validity of a new measure of functional mobility. *Pediatric physical therapy : the official publication of the Section on Pediatrics of the American Physical Therapy Association*, 16(2):90–98, 2004.

[Zon] Intel Developer Zone. Character Animation: Skeletons and Inverse Kinematics - https://software.intel.com/en-us/articles/character-animation-skeletons-and-inverse-kinematics.