FORCE FEEDBACK THERAPY FOR CEREBRAL PALSY PATIENTS

BY

BAS GEERDINK

University of Manchester, 2003
University of Groningen, 2004

17 March 2004

THESIS

Submitted in partial fulfillment of the requirement
for the degree of Master of Science in Artificial Intelligence
in the Graduate College of the
University of Groningen, 2004
under supervision of
Dr. F. Cnossen and Dr. R. Richardson

Acknowledgements

First, I want to thank my parents for giving much support throughout my whole study time. They always supported my choices, and stimulated me to move abroad.

Thanks to his humour and likeliness to me, my brother Stan was the sober background force during my time in Groningen and Manchester.

I would like to thank my supervisors Fokie Cnossen and Robert Richardson for giving me guidance and invaluable advice. Up to the time I finished this thesis, I could always count on their time and support. Next to their scientific experience, they were fantastic people to work with.

Many thanks go out to Martin Levesley who introduced me to Leeds University and invented a magnificent chair for the children. He also supplied lots of fancy stuff, such as a video camera.

I want to thank Bipin Bhakta, Stefan Spinty, Dawn Blackaby and Rajini Sarvananathan of St. James Hospital in Leeds for their enthusiastic cooperation, providing the test subjects, and hosting the joystick experiments. Because of them, my time in Manchester was very pleasant, and the implementation and testing of the software went smoothly.

Finally, I want to thank all my friends both in England and in Holland for keeping me motivated throughout the project. During my time in Manchester, I met many good friends who made my stay very pleasant. Still, the contacts with my friends back home were so intensive that I longed back to Groningen many times.

968

Table of Contents

List of Figures

Chapter 1

Introduction

Cerebral palsy (CP) is the term for certain disorders of muscle movement and body posture that have occurred as a result of an injury to the brain [19]. Applying physiotherapy can help people with this syndrome. With practice, children can tremendously improve their eye-hand coordination [11]. Unfortunately, this is a very costly process in both money and time aspects.

Some hospitals have machines available that can assist the physiotherapist, but those are not suitable for people in their own home environment since they are too big and too expensive. Treatment of CP would be a lot easier if people suffering from it could perform exercises themselves without the aid of a professional therapist. To reach that goal, a *force feedback* joystick (see figure 1.1) may be beneficial for the children. Mass-market computer games frequently use normal joysticks; force feedback is relatively new. A joystick with this feature can exert force, such as vibrations, to its handle. In that way, the person holding the joystick gains extra game-feedback besides audiovisual information. The latest force feedback joysticks can push their own handle to any direction, so the user experiences assistance or resistance to joystick movements.

Figure 1.1: A force feedback joystick: the Microsoft Sidewinder 2

Our research project aimed at finding a new therapy type for CP children. Standard physiotherapy is useful for improving movement range and controlling CP effects such as spasticity, but it is expensive and not entertaining for the children. Since force feedback can help directing the movement of children, we designed and implemented a computer game that was controllable by a force feedback joystick.

We tested the software by performing two experiments. In the first experiment, healthy test subjects played the game without force feedback help. In the second experiment, we tested the reaction of children with CP on a force feedback game. Analyzing the data from the games, as well as elbow angles from a goniometer, we determined if children with CP trained their arm movements by playing the game with force feedback assistance.

This Master thesis will discuss issues such as the theoretical background of the project (chapter 2), and the method (chapter 3). In chapter 4, we will present the test results, and chapter 5 contains a summary of the project, the overall conclusions, and suggestions for further research. We summarize technical issues such as software implementation in Appendix A to D.

Chapter 2

Background

This chapter describes the physical attributes of the test subjects, possible treatments of CP, previous attempts to discover new therapy types for CP, and the reason we chose to perform research with a force feedback joystick.

2.1 Cerebral Palsy

Ingram [5] published the following definition: 'Cerebral palsy is an inclusive term used to describe a number of chronic, non-progressive disorders of motor function, which occur in young children as a result of disease of the brain.' CP occurs two to three times in every thousand live births.

2.1.1 Causes

Causes of CP are brain lesions or maldevelopment of the brain before, during, or shortly after birth; diseases of the mother during pregnancy; and brain damage as a result of choking, poisoning or near drowning. The effect is abnormal motor development, which can lead to a spastic child [3]. Lance [7] defines spasticity as 'a motor disorder characterized by a velocity dependent increase in tonic stretch reflexes (muscle tone) with exaggerated tendon jerks, resulting from hyper excitability of the stretch reflex as one component of the upper motor neuron syndrome'. Besides spasticity, another form of CP exists, called athetoid CP, in which patients suffer from unpredictable movements and often have weak muscles.

2.1.2 Diagnosis

It is difficult to diagnose CP in babies, because there are not many clear indications of the disease. A doctor can however look for symptoms of CP, such as the lack of the neck righting reaction (i.e. the head rotation of a baby lying on his or her back is followed by rotation of the body as a whole), and the lack of a postural tone against

gravity (for example, the head of a child falls behind when he is being pulled up from sitting).

After about six years, CP often results in spasticity, which mostly shows as stiffness or tightness of muscles. Spasticity is one of many types of CP, and can occur in three forms, possibly combined [3]: in spastic diplegia or paraplegia, the disease typically affects the lower extremities of patients, resulting in crossed legs (scissors posture) and walking difficulties. Spastic quadriplegia affects all four limbs of the body, but the distribution is mostly asymmetrical. Spastic hemiplegia affects one side of the body, which often results in children doing every action with the healthy side. Other handicaps such as speech impairment and seizures (attacks of convulsive movements) sometimes result in psychological or behavioural problems. For instance, some children with CP cannot attend normal schools because of their physical handicaps. On special schools, they often feel extra handicapped, and may be limited in mental development [4]. Therefore, CP often goes together with mental retardation.

At the age of ten to twelve years, CP children will typically experience involuntary or difficult movements, and disturbances in gait and mobility [17]. Arm movements in space are less smooth and stable than of healthy people. Compared to normal children, qualitative adjustments of initiated movements are equally appropriate; a CP child can change an initiated movement with the same speed and accuracy as a healthy child. There exist three groups of disorders [1]:

- disturbances in muscle activity, divided in negative effects (loss of muscle power and coordination), positive effects (involuntary muscle contractions such as cocontractions of opposing muscles), and other effects (e.g. abnormal reactions to skin stimulation);
- disturbances in muscle stiffness: decreased flexibility of muscles because of changed mechanical properties;
- disturbances in muscle length: sometimes muscles shorten because of CP.

CP can affect brain areas such as the thalamus, which among other things regulates information transport between muscles and brain areas. Disorders of movement initiation can be the result of damage to cerebral motor areas.

2.1.3 Summary

Generally, a CP child of approximately ten years old has difficulties controlling one or more limbs in a natural way. Therefore, he wears bracers around joints that are affected by the disease, and does exercises every day at home under supervision of a physiotherapist, who checks the progress every week. Once or twice a year, the child will usually attend a CP clinic in the hospital. If necessary, the child will receive a botulinum toxin injection (often referred to by the product name 'Botox') into stiff muscles to weaken them, and thereby improving the body posture. We will discuss the mostly used therapy types in section 2.2.

2.2 Treatment of Cerebral Palsy

There are several ways of fighting CP. The sort and amount of treatment patients receive depend on factors such as age and the amount of influence that the disease has on everyday activities. CP treatment can be divided into surgery, medication, and physiotherapy. We will discuss each of these treatments. Notice that because of many different causes and symptoms as described in section 2.1, CP is difficult to fight. Doctors across the world have a variety of approaches to attack the disease. Some will emphasise medication, while others have to rely on physiotherapy because of the lack of good medicines. In Europe, there seem to be enough money available for extensive treatment. Therefore, patients across Europe often receive a combination of various curing methods.

2.2.1 Surgery

Applying surgery to a mature CP patient can reduce the effects of the handicap. The thalamus is the brain structure that transmits information from the muscles and sensory organs to other parts of the brain. In an adult with severe CP, cutting parts of the thalamus by a neurosurgeon can reduce spasticity, but it is a risky procedure. On the other hand, Orthopaedic surgery treats the muscles directly instead of the brain [6]. The goal of muscle surgery is to obtain functional positioning of the arms, thereby consequently reducing spasticity and restoring movement functions such as grabbing objects. For instance, a healthy human arm motion initiates by contracting certain muscles and simultaneously extracting other ones. CP patients often cannot

make this synchronized movement and contract all involved muscles, resulting in no arm movement at all, or spastic jerks of the arm. Surgery of the forearm can correct this by muscle transference: surgeons lengthen certain arm muscles to gain more freedom of movement.

2.2.2 Medication

Treatment of CP children also includes medication [10], allowing patients to live in a more manageable way. The most often used medicine is Botox, which is injected into stiff muscles. Bones and muscles of children up to sixteen years old grow at a large speed. In CP, the body sometimes cannot match the growth of separate parts of an arm or leg, resulting in incorrect muscle lengths. An injection in a short muscle will loosen it, allowing patients more movement freedom despite incorrect muscle growth.

2.2.3 Physiotherapy

Physiotherapy, in the form of individual physical training, is crucial for improvements in motor control. A frequently used therapy treatment program is the Bobath technique, in which therapists counteract primitive CP reflexes by forcing opposed movements in children [12]. Some other examples of therapy types [13] are:

- Sensory Integration: based on the hypothesis that input disorders from the sensory system cause movement malfunctioning;
- Neurodevelopment Therapy: based on the idea to let children move through stages of development by encouraging correct movements and discouraging incorrect or primitive reflexive ones;
- Root Method: based on body stimulating, e.g. touching the bottom of the foot with heat or cold should cause different reactions in children compared to adults.

In the next session, we will first look after previous attempts to alter CP treatment.

2.3 Previous Research

All traditional therapy types for CP require intervention of a physiotherapist. These therapies work, but only if a therapist spends time and money on each individual child. A new approach to CP therapy has come with the computer revolution. It is now possible to let a computer take over some tasks from a physiotherapist.

2.3.1 CP Therapy

Over the last ten years, several research projects have investigated new CP therapy types. Van den Berg-Emons [2] wrote a thesis about physical training studies on children with CP. She states that by aerobic sessions patients can accomplish significant gains in aerobic power and muscle strength as well as a reduction of spasticity. She found no effect on the mechanical efficiency of test subjects.

Thorpe *et al.* [18] developed a ten-week program in which nine spastic diplegic CP patients performed aquatic exercises. They improved strength, balance, functional mobility, and self-perception.

Another therapy possibility is telerehabilitation. Lathan [8][9] describes a robot for disabled children, which operated by the child's voice and body movements. The robot, which looked like a furry stuffed animal, mimicked the actions of the child. The robot could function in a classroom, so the child could attend classes through the robotic interface. Disabled children could push their motion limits by receiving rewards like a funny robot-dance when showing behaviour they are supposed to learn. These studies show that fighting CP with a well-designed exercise program is possible, and that improved rehabilitation methods can be discovered.

2.3.2 Stroke

Stroke can lead to similar symptoms as CP, for example spastic paraplegia. Hence, it is interesting to examine a project on post-stroke patients. Reinkensmeyer *et al.* [14]–[16] performed a research project in which he used a force feedback joystick to test patients in a Java-based computer game, played over the Internet. His results showed that a force feedback joystick is helpful for movement practice of stroke patients. Test subjects showed an improved movement precision, and movement speed with the game increased by 40%. This project stimulated us to invent a CP therapy type based on a force feedback joystick.

2.4 Current Research

Despite all previous research, no one has yet discovered a good replacement for standard physiotherapy. We wanted to find a way to let CP children perform entertaining exercises that train them in a similar way as a normal physiotherapist would, because the results of traditional physiotherapy are generally good. Since physiotherapy centres on practicing movements by means of simple exercises, our aim was to invent a training program that included similar movement exercises, which were entertaining but presented the same kind of tasks to children with CP.

2.4.1 Force Feedback

We believed that a force feedback joystick would be a helpful tool to help the children. The advantages of such a joystick in comparison with other devices such as professional hospital equipment are the relatively low price of the hardware and the ability to install it on almost any computer. At the start of this research project, we did not know all the potential and drawbacks of the joystick. Therefore, in chapter 3 we will give an extensive description of joystick measurements, including force feedback output. To study all the options of the joystick, two experiments were set up. The first experiment only tested healthy subjects, who did not receive force feedback help from the joystick. In the second experiment, test subjects with CP worked with the full possibilities of the joystick. By examining the results of the two experiments, we wanted to get a comprehensive view of the potential of the joystick as a CP therapy device.

Our objective was to let the joystick help the children by assistive force feedback. That is, the joystick pushed the arm of the children towards a desired goal position. We implemented assistive pushes from the joystick to let the children move their arms more easily, especially if the children had difficulties to initiate movements towards a desired position. When, because of CP, spasticity occurred, or the child could not move his or her arm in a desired direction, the joystick provided a stimulant that helped the child directing his initial movement.

2.4.2 Test Subjects

The children who we worked with in Leeds performed exercises every day themselves. Once or twice a week, a physiotherapist who examined the progress and

if needed prescribed new exercises, visited them. Additional, they attended a clinic in the hospital in which staff members examined them twice a year. Doctors closely recorded the progress of movement freedom throughout the entire childhood of the patients. Patients we interviewed were content with the current treatment methods, but were happy to cooperate to a research project that would possibly make therapy more entertaining and minimize the influence of physiotherapists. We clearly understood that CP patients could benefit from a new approach to standard treatment

2.4.3 Research Question

The overall goal of this project was to investigate the possibility to help spastic CP children by means of a computer program and a force feedback joystick. Our research focused on the construction of a computer software interface that was affordable and easy to install. Our research question was whether the rehabilitation of children with CP would benefit from short exercises with a force feedback joystick interface. This involved the design of the computer program, the amount and sort of help the joystick provided, and the mechanical modifications of the work environment. The end-goal of the interface was rehabilitation of children with CP. Therefore, we had to perform experiments in which both healthy and CP-infected test subjects had to play the computer game, and data were gathered about performance and movement characteristics. Afterwards, we analyzed those data and tried to reach a conclusion about the usability of the software and the amount of progress the children made when receiving force feedback help from the joystick.

Chapter 3

Method

As mentioned before, we performed two separate experiments. This chapter describes the test subjects and the software we used. To comprehend the situation of the test subjects during testing, we studied the physical attributes of the joystick. A chair with a joystick platform created an adjustable working place for the test subjects.

3.1 Test Subjects

To make use of the maximum movement range of the arms of the children, we decided to extend the shaft of the standard joystick. Because the experiments tested children of several ages, we constructed two joystick extensions of different lengths. Small children could use the short extension, tall children the long one. The extensions were easily replaceable. Section 3.4 describes the joystick extensions in full.

In the first experiment, we tested the hard- and software without force feedback on five healthy children. The test group consisted of two girls of four years, one boy of eleven, and a boy and a girl of thirteen. Only the two boys had previous experience with computer games and joystick play. The two four-year old girls used the short joystick extension; the other children used the long one.

In the second experiment, we tested the game on three children with CP: a boy of seven years old, a girl of seven, and a girl of nine. All these children had movement difficulties with their right arm. The girl of seven used the short joystick extension; the others used the long one.

3.2 Experimental Tasks

The largest part of the project was the design and implementation of the test software. Two separate experiments required two different games. In the game for the first experiment, we did not use force feedback output yet.

3.2.1 Computer Games

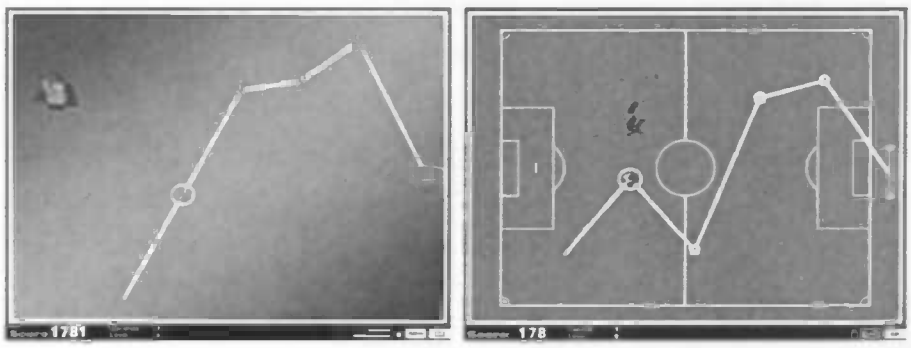
The game program was a full-screen computer game, running in Microsoft Windows XP on a laptop. The game interface consisted of a laptop connected with a Microsoft Sidewinder 2 Force Feedback joystick through a USB port. The goals of the game for the first experiment were to test the overall performance of healthy test subjects and examine their movement patterns. For this experiment, to let the children play in a friendly atmosphere we decided to build a game around a monkey that searches for bananas. The control object was a monkey that could move in four separate directions: left, right, up and down. The target objects were bananas, which appeared at random positions on the screen. When the monkey was moving, animation showed on the screen to give the game a feeling of a commercial computer product.

We included several changeable game settings, such as target highlighting, which made a circle appear around the target object that faded from yellow to red, to make the bananas more visible. Another selectable setting was moving targets, which added some difficulty in capturing the bananas by making the move slowly in a random direction across the screen. There were two possible relations between the joystick and the control object, which are described in section 3.2.2. Next to these different movement types, we constructed two game types, which are described in section 3.2.3. All settings, movement type, and game type were selectable on a separate window, which popped up when the game started or a button on the screen was clicked with the mouse. The score of the test subjects was visible on the screen, as well as a level count. After capturing ten bananas, the next level commenced with a different background picture. Figure 3.1 shows some action screen shots of the game that we used in the first experiment. For an extensive description of this computer game, see appendix A.

(A) (B)

Figure 3.1: Two screenshots of the monkey game that was used for experiment 1: a basket game (A) and a traject game (B)

For the second experiment, we decided to change the appearance of the game. We constructed a football game, which was identical to the monkey game except for the graphical interface. The children could choose between the two games, which added to the attractiveness. Screen shots of this game are depicted in figure 3.2. The tasks of the test subjects remained the same, with the addition that force feedback assistance was available. We give a description of the force feedback output function in section 3.2.4. Appendix B contains an extensive description of the game for the second experiment.



(A) (B)

Figure 3.2: Two screenshots of the games used for experiment 2: a monkey game (A) and a football game (B). Both are traject games.

3.2.2 Movement Types

In the computer, it is possible to translate any position of the joystick handle into horizontal X and vertical Y coordinates. There are several ways to translate the

joystick position to the position of the action figure on the screen. A few aspects of the joystick come to mind. First, the movement range of a joystick is square, while a laptop screen is rectangular in the ratio of approximately 4:3. Second, traditional computer games that are controllable with a joystick generally use an acceleration mechanism for the action figure, in which the angle of the joystick determines the amount of speed in which the action figure moves. We decided to implement two different movement types. In the *velocity based* movement type, the speed and direction of the movement of the action figure are dependent on the joystick angle. The further test subjects would push the joystick towards its extreme angles, the faster the monkey on the screen would move in the direction of the joystick. In the *position based* movement type, we made a direct connection between the joystick position and the place of the monkey on the screen. Therefore, when test subjects pushed the joystick towards its extreme left-forward position, the monkey would instantly move to the top-left position on the screen.

3.2.3 Game Types

There were two different game types requiring different kinds of arm movement patterns: the *basket* game type and the *traject* game type. In *basket* games, used for initial practise with the joystick, the test subjects had to make a reciprocating movement from the centre to a random position on the screen, where the picture of a banana was displayed. We used *traject* games to actually test the children. The test subjects had to follow a trajectory across seven points on the screen, of which five had randomized Y coordinates. The trajectory always followed a path from the left to the right of the screen.

3.2.4 Force Feedback

The second experiment, with CP patients, made use of force feedback output. Force feedback was adjustable by making use of two variables on the options screen: *level* and *dead zone*. The force feedback, if enabled, followed a linear function that depended on these variables. We related the amount of force output directly to the absolute distance on the screen between the control object (monkey or football player) and the target object (banana or football). Test subjects received force feedback in a linear relation to that *error*, stated in equations 3.1 and 3.2. There are two identical functions, for force feedback in X and Y directions.

$$\text{PowerX} = \begin{cases} 0 & \text{if } |E_x| < DZ \\ -(|E_x| - DZ) * FL & \text{if } E_x < 0 \\ (|E_x| - DZ) * FL & \text{if else} \end{cases} \quad (3.1)$$

$$\text{PowerY} = \begin{cases} 0 & \text{if } |E_y| < DZ \\ -(|E_y| - DZ) * FL & \text{if } E_y < 0 \\ (|E_y| - DZ) * FL & \text{if else} \end{cases} \quad (3.2)$$

In these equations, E_x and E_y denote the errors in X and Y directions, that is the distances between the target object and the control object. The *DZ* variable denotes the *dead zone*, which was adjustable by the program user. The range of *DZ* was from zero to 125 in steps of five. *FL* denotes the force feedback *level* variable that was also adjustable by the user. Its range was from –50 to 50, with negative values allowing resistive instead of assistive force feedback output.

The first line of each equation results in the actual dead zone of the joystick. The *Power* output in X or Y direction will be zero if the absolute error in that direction is smaller than the dead zone setting *DZ* allows. The second line of the equations sets *Power* to a negative value if the corresponding error is negative as well, so the force points in the correct direction. The rest is, as mentioned, a linear function: *Power* increases with growing error and with higher force feedback level *(FL)* settings. The maximum *Power* setting of the software was 10000. Any results of *Power* calculations with equations 3.1 and 3.2 that exceeded that maximum were set to 10000. The output graph of the force feedback is depicted in figure 3.1.

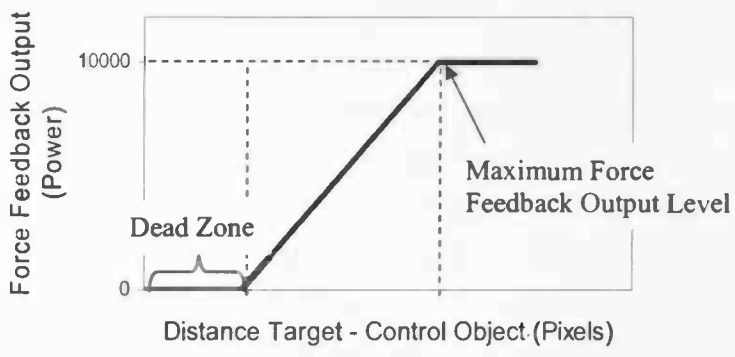


Figure 3.3: The relation between the distance control object – target and the amount of force feedback output

We discuss the relation between *Power* and the actual force output of the joystick in Newton in section 3.4. We implemented the program in Borland Delphi 5. Appendixes A and B contain extensive descriptions of the source code.

3.2.5 Summary

In the first experiment, we tested five healthy children with version 1 of the joystick game. The goal of that game was to make a monkey run across the screen and capture as many bananas as possible. All children practised two different movement types for controlling the monkey: a velocity based, and a position based relation between the joystick and the monkey on screen. Besides that, the children performed two kinds of game types. Some basket games were played, but most sessions were traject games. We experimented with several other game settings such as moving targets and time intervals within which the target had to be reached.

Three children with cerebral palsy performed the second experiment. The test subjects chose between two similar game variants: the monkey game or a football game. Each test session started with one basket game with velocity based movement for practising. The rest of the test games were traject games with position based movement. We varied the force feedback output and the dead zone range across the game trials. Section 3.3 describes the test procedure in full.

3.3 Procedure

Tests of the second experiment took place in a private room in St. James Hospital. At least one parent, who was present during the whole test session, accompanied the children. Children and parents had to sign a consent form before joining the tests. When the child with the parent entered the room, we fully explained the experiment to them and gave a demonstration of the software. It was important to create a comfortable working environment for the test subjects, so we supplied food and drink, and gave them enough time to ask questions. After that, the children took place on the adjustable chair and we attached a goniometer to the child's elbow.

When everything was set up, we started with the tests for experiment 2 in series of two minutes. In between the games, the children got three minutes rest. After five two-minute games, the children got a half hour rest, after which another series of five two-minute games started.

The test sequence of the first two children was two times five games in the following order: no force, low force, high force, low force, no force. This 'peak' sequence was used because we wanted to test the influence of the force feedback, and after practise, the children should perform better. The children had three minutes rest in between the trials, and half an hour after the first five games. Figure 3.4 displays this time sequence for one out of two test sessions.

	Duration in minutes	*Game Type*	*Movement Type*	*Force Feedback level*	*Dead Zone range*
warm-up	2	basket	velocity	0	–
rest	3				
trial 1	2	traject	position	10	10
rest	3				
trial 2	2	traject	position	20	5
rest	3				
trial 3	2	traject	position	10	10
rest	3				
trial 4	2	traject	position	0	–
rest	30+				

Figure 3.4: Time sequence of test sessions in experiment 2 for test subjects 1 and 2. This sequence was repeated once. Note the peak-shaped force feedback distribution.

The test sequence of the third child was: no force, no force, low force, low force, high force. In this way, we could investigate the influence of fatigue effects on the children when we compared the performance of all the children. Figure 3.5 depicts the time sequence for test subject 3 for one out of two test sessions.

	Duration in minutes	*Game Type*	*Movement Type*	*Force Feedback level*	*Dead Zone range*
warm-up	2	basket	velocity	0	–
rest	3				
trial 1	2	traject	position	0	–
rest	3				
trial 2	2	traject	position	10	5
rest	3				
trial 3	2	traject	position	10	5
rest	3				
trial 4	2	traject	position	20	10
rest	30+				

Figure 3.5: Time sequence of test sessions in experiment 2 for test subject 3. This sequence was repeated once. Note the increasing force feedback distribution across the trials.

3.4 Apparatus

We used a Microsoft Sidewinder 2 joystick, connected to the computer through a Universal Serial Bus (USB) port. It communicated with the laptop by Windows API DirectX software. The joystick was extended to create more movement freedom for the children (see figure 3.6). The two extensions consisted of plastic tubes, which were placed on the existing inner shaft of the joystick. A ball-shaped handle at the end of the tubes allowed the children different options of gripping the joystick.

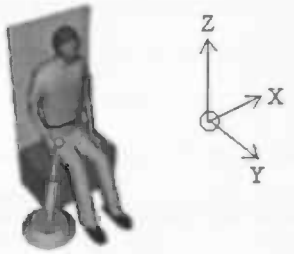


Figure 3.6: A seated person operating an extended joystick. In anatomical terms, the X/Y plane is transversal, the X/Z plane is frontal, and the Y/Z plane is sagital.

The total movement angle of the joystick was 35 degrees. The short extension was 30 centimetres in length, resulting in a movement range of 21 centimetres in horizontal and vertical directions. The long extension of 42 centimetres resulted in a movement range of 28 centimetres in both directions (see figure 3.7).

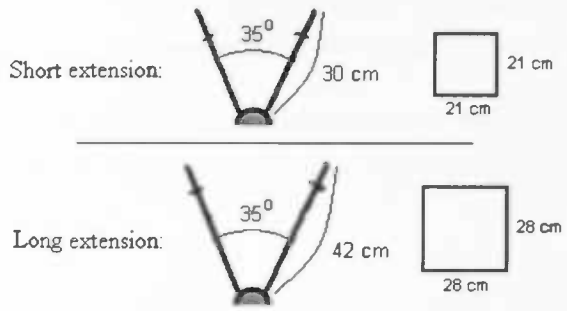


Figure 3.7: Joystick measurements

We measured the absolute force feedback output in Newton by attaching a spring to the ends of the two joystick extensions. First, we determined the spring

constant k by fastening weights to the spring and measuring its stretching as result of gravity.

$$F_{spring} = -F_{gravity} \quad (3.3)$$

$$k * \Delta l = -m * g \quad (3.4)$$

In formula 3.4, k[N*m^{-1}] is the spring constant, l[m] is the spring extension, m[kg] is the mass that hangs under the spring and g[m*s^{-2}] is the gravitation constant. Since g is 9,81 in England, we could calculate k by varying m and measuring l. By averaging five measurements, we estimated k at 2,70.

We determined the force output of the joystick by varying the *Power* variable of the DirectX joystick program from 0 to 10000 in steps of 1000 (see figure 3.8). The *Power* value of 10000 was the maximum force output setting of the software.

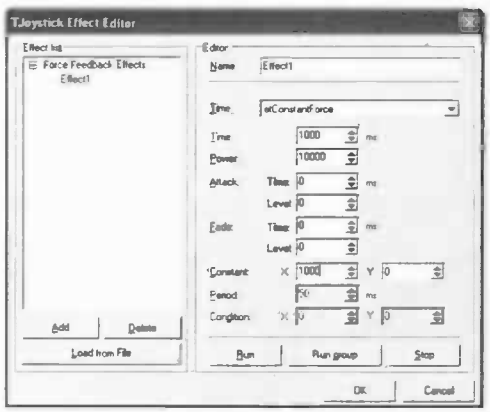


Figure 3.8: The list of adjustable force feedback effects. One of them is *Power*, which sets the force output from 0 to 10000.

The force output had to be measured for the two different extension lengths of the joystick, 30 and 42 centimetres. We calculated the force with formula 3.5.

$$F_{spring} = k * \Delta l \quad (3.5)$$

We measured the force in three joystick positions (see figure 3.9): when the joystick pulled the spring from its neutral resting position (1), when we moved the spring back, so that the joystick returned to its original position but the spring was still extended (2), and when the spring was pulled even further back, so that the joystick is angled towards the other end that the force is pulling it (3). It turned out

that these joystick positions were of no influence of the force; in all situations, the spring length was equal.

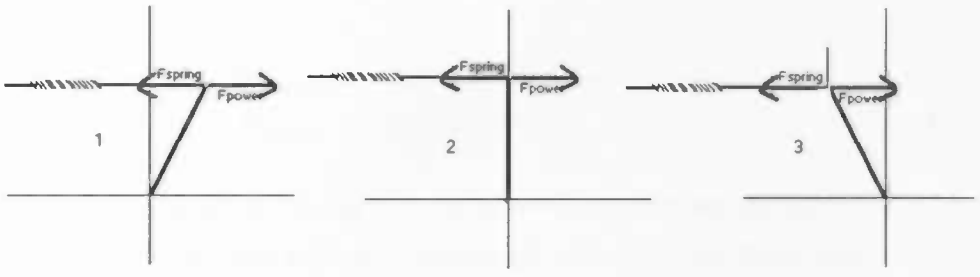


Figure 3.9: Joystick positions while measuring force output. F_{spring} is the force of the spring; F_{power} is the force feedback that pulls the joystick in the opposite direction.

The results of these measurements are depicted in figure 3.10.

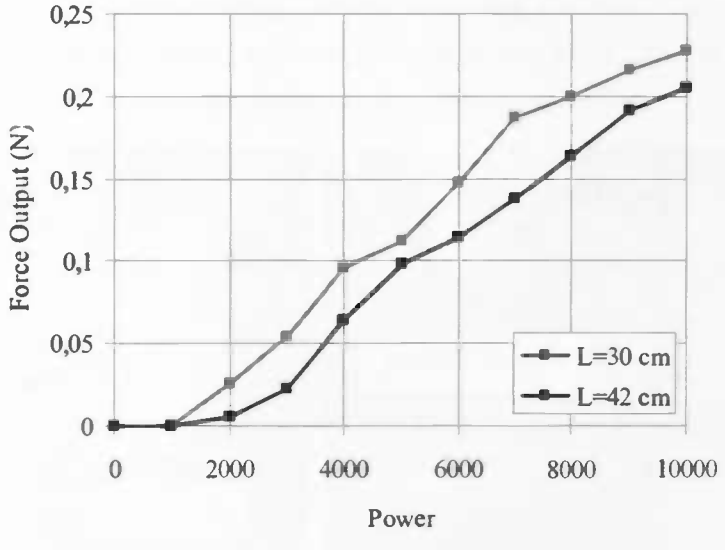


Figure 3.10: The measured force of the joystick when the Power setting of the DirectX software is varied from zero to 10000

To conclude, the force increases linearly with the *Power* variable. This is important for the game because it clarifies the effect that the value of *Power* has on the user. When *Power* increases or decreases with a certain amount, the user will experience a similar increase or decrease in the assistance or resistance that the joystick provides. In addition, the graph shows that the force is larger when the

extension is short. That effect follows from the law that force times length is always equal. Finally, at a *Power* level of 1000 there is no measurable force. Therefore, the Force Feedback approximately follows formulas 3.6 and 3.7, directed from figure 3.10.

$$\text{if } L = 30: \quad F_{joystick} = 0.23/9000 * (Power - 1000) \tag{3.6}$$

$$\text{if } L = 42: \quad F_{joystick} = 0.21/8000 * (Power - 2000) \tag{3.7}$$

In these formulas, $F_{joystick}$[N] is the actual force that the joystick exerts to its handle and *Power* is the force feedback output of the software, which ranges from 0 to 10000.

The working area of the CP children during the tests consisted of a desk, a laptop with the software implemented on it, an extended force feedback joystick, a foot bench, and an ergonomic chair. We positioned the joystick in front of the children, to let them train a movement that is similar to grabbing objects. The chair was adjustable in height for creating a comfortable sitting position. It had a movable platform, on which we attached the extended joystick (see figure 3.11). In that way, we could determine the optimal working position for each test subject.

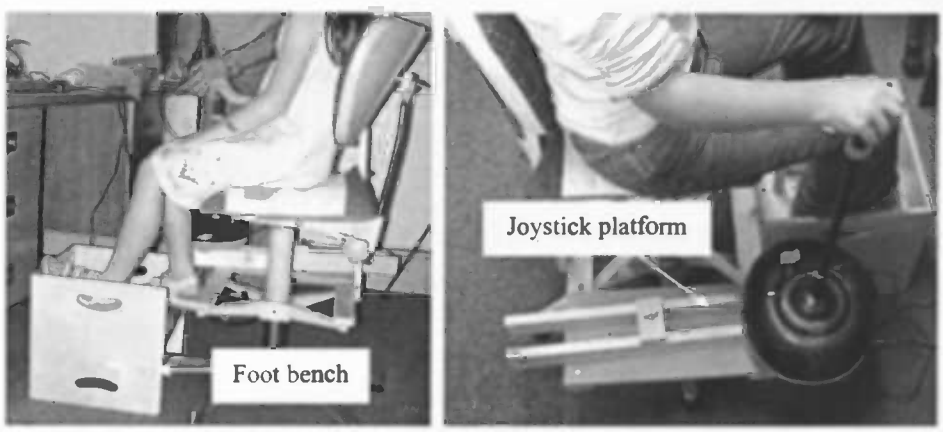


Figure 3.11: Two photographs of test subjects sitting in the adjustable chair

3.5 Data Collection

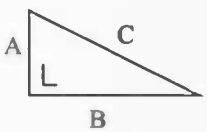
Our software provided several data variables, which we recorded at run time. There were two different kinds of data: in-game dynamic data variables that were stored every fifty milliseconds, and general static game information. The end-scores of the

test games were the most important stored data variables, because they indicated how much movement the children had made within the duration of the game. The end-score was the sum of all absolute distances between the control object and the target objects, measured in screen pixels. For instance, if in one traject game a test subject had completed five trials with seven target objects each (five bananas plus the start point and the basket), the total end-score would have been the sum of twenty-five calculations of the distance between the control object and the next target. The computer software made those calculations at the time the test subject reached the target. We used the formula of Pythagoras (see equation 3.8) on X and Y distances to calculate the absolute distance between the objects.

3.5.1 Equations

Calculations of data values made use of the following equations.

$$C^2 = A^2 + B^2 \tag{3.8}$$



$$T_x(t) = |XControl(t) - XControl(t - 50)| \tag{3.9}$$

$$T_y(t) = |YControl(t) - YControl(t - 50)| \tag{3.10}$$

$$V_x(t) = |XControl(t) - XControl(t - 50)| / 50 \tag{3.11}$$

$$V_y(t) = |YControl(t) - YControl(t - 50)| / 50 \tag{3.12}$$

3.5.2 Dynamic Data

Every fifty milliseconds, the game wrote a line to a text file called 'XYData'. That file contained information about the game situations during the tests. Twelve data variables were stored for each game:

1. the time of the measurement, in units of fifty milliseconds;
2. the horizontal (X) position of the joystick, which ranged from 0 to 1000;
3. the vertical (Y) position of the joystick, which ranged from 0 to 1000;
4. the horizontal position of the control object on the screen in pixels;
5. the vertical position of the control object on the screen in pixels;
6. the horizontal position of the target object on the screen in pixels;
7. the vertical position of the target object on the screen in pixels;
8. the error in horizontal direction in pixels, which was the difference between horizontal positions of the control object and the target object;

9. the error in vertical direction in pixels, which was the difference between vertical positions of the control object and the target object;
10. the total error in pixels, which was the absolute distance between the control object and the target object, calculated with the formula of Pythagoras;
11. the force feedback output in horizontal direction, which ranged from 0 to 10000;
12. the force feedback output in vertical (Y) direction, which ranged from 0 to 10000.

Figure 3.12 shows an example of a dynamic data file. There are 2400 data lines, because during the games of two minutes the software wrote a line to the text file every fifty milliseconds.

```
Time(*50ms) JoystickX JoystickY XControl YControl XTarget
        YTarget XError YError TotError ForceX ForceY
1 -1000 14 16 312 204 467 188 155 244 6520 5200
2 -995 26 16 316 204 477 188 161 248 6520 5440
...
2400 -77 51 407 324 371 95 -36 -229 232 -440 -8160
```

Figure 3.12: Example of 'XYData.txt' file, in which dynamic game data was stored

After the tests, we analyzed the twelve recorded data variables and performed some calculations on them. To gain more insight in the movement behaviour of the test subjects, we extended each data line with another twelve variables:

13. the absolute distance in horizontal direction that was travelled by the control object in the last fifty milliseconds, in screen pixels. This was calculated by using equation 3.9, in which $T_x(t)$ is the travelled horizontal distance on time t;
14. the absolute distance in vertical direction that was travelled by the control object in the last fifty milliseconds, in screen pixels. This variable was calculated by using equation 3.10.
15. the velocity in horizontal direction, in screen pixels per millisecond. We calculated the velocity by using equation 3.11, in which $V_x(t)$ is the velocity in horizontal direction on time t. The variable gives the absolute average speed in horizontal direction of the last fifty milliseconds;
16. the velocity in vertical direction, calculated by using equation 3.12;
17. a variable that checked if a new trial had started in which a new target object was placed on the screen;

18. the absolute error distance in horizontal direction of the beginning of each trial, in screen pixels;
19. the absolute error distance in vertical direction of the beginning of each trial, in screen pixels;
20. the total horizontal cumulative distance travelled in each trial;
21. the total vertical cumulative distance travelled in each trial;
22. the extra horizontal distance travelled on the last trial. This is the travelled X distance reduced by the absolute horizontal distance at the start of the trial;
23. the extra vertical distance travelled on the last trial. This is the travelled Y distance reduced by the absolute vertical distance at the start of the trial;
24. the total extra distance travelled on the last trial, calculated using the formula of Pythagoras on variables 22 and 23.

3.5.3 Static Data

Next to the dynamic data files, game settings and information about the test subjects was stored in text files called 'EventData'. Those files contained the following data:

25. the name or number of the test subject;
26. the date of the game;
27. the starting time;
28. the end time;
29. the number of the game;
30. the duration of the game;
31. the movement type (position/'XY' based or velocity based, see section 3.2.2);
32. the game type (basket or traject, see section 3.2.3);
33. movement of the target object ('MB': yes or no);
34. highlighting of the target object ('Circle': yes or no);
35. visibility of the trajectory ('Traject': yes or no);
36. time interval (0 to 25);
37. force feedback level (-25 to 25, see section 3.2.5);
38. force feedback dead zone (0 to 25, see section 3.2.5);
39. time and score (travelled distance 'Dist') of completed trials;
40. the total number of trials that the subject played within the game duration of two minutes;

41. the end-score, which was a measurement of the total cumulative distance between the target objects.

The two force feedback data variables were not recorded in the first experiment, because force feedback was not yet implemented in the version of the computer game that we used in that experiment. Figure 3.13 displays an example of a static data file. The number of recorded lines in the text file depended on the number of trials that the test subject completed.

```
Test subject 2, 12-10-2003, game nr. 9
0 : INIT --> Started game with duration of 2 min, Basket,
        XY, FF: level 10, DZ: level 20, MB, Circle, Interval
        level 15, NO Traject.
27 : Completed trial 1 with Dist 169.
54 : Completed trial 2 with Dist 320.
...
2377 : Completed trial 5 with Dist 421.
2400 : END --> total score = 24255
```

Figure 3.13: An example of 'EventData.txt' file, in which static game data was stored

Based on the 'extra' dynamic data variables 13 to 24, we calculated four static data variables for each game:

42. the average time the completion of one trial took, which is the number of trials divided by 2400 milliseconds (two minutes);
43. average extra horizontal distance travelled: the value of variable 22 divided by the total number of trials;
44. the average extra vertical distance travelled: the value of variable 23 divided by the total number of trials;
45. the average extra total distance travelled: the value of variable 24 divided by the total number of trials.

3.5.3 Elbow Angles

In the second experiment, we used a goniometer (see figure 3.14) to measure the elbow angle of the children while they were playing the joystick game. The goniometer consisted of two prism-shaped magnets with a spring in between them. We attached he magnets to two sides of the elbow joint with a piece of tape. A cable connected the goniometer to an electronic data plotter. After each trial, we read the data from the plotter into the laptop in the three-minute pause that the children

received. Those data were stored in standard text files with elbow angle values (see figure 3.15).

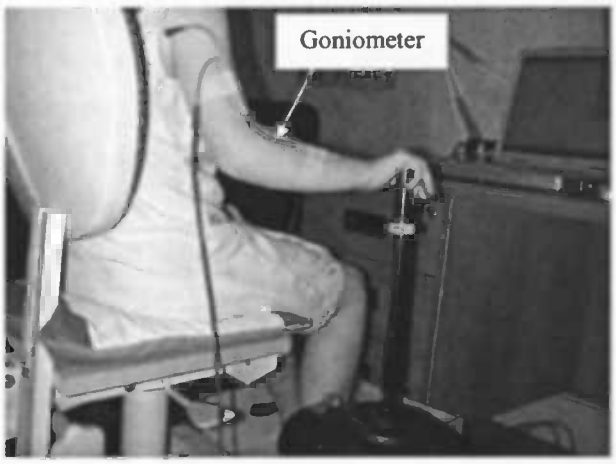


Figure 3.14: A child holding the joystick with her arm attached to a goniometer

```
Sampling Rate: 20 Samples/Sec
Number Of Channels: 1
Ident Number:  0
Channel A Cal1: 135 Cal2: 199
Total Number Of Bytes Recorded: 65536
Ch.A
65
63
61
...
71
```

Figure 3.15: An example of a data file from the goniometer. The first five lines contain information from the data processor. From line 6, each line contains an elbow angle value.

To summarize, at the end of each game played by the test subjects we got three text files: one with dynamic game data, one with static game data, and one with elbow angles from the goniometer. Chapter 4 describes how we processed this information.

3.6 Human Arm Simulation

To let a physiotherapist evaluate the movements of the children, we designed and implemented a dynamic human arm movement simulation. The program can play back movements of the test subjects in a virtual room after loading a 'XYData' file

with dynamic game data from the joystick game (see section 3.5.2). The simulation sets the joystick at the same position as in the data file, and displays the joystick together with the arm of the test subject on the screen. The simulation is in OpenGL, and the position of the arm is calculated and viewed on screen in three-dimensional space. We included several options in the simulation, such as the possibility to set the view angle by virtually walking across the room.

The only assumption we made is that the shoulder of the test subjects was set at a given point in space. That is incorrect because the shoulder always tended to move as the children played the game, but it makes predictions about the position of the elbow possible. By looking at the simulation, a physiotherapist can get a good idea of the movements the children have made while practising. It could become a useful tool for people who have to estimate the progress of CP patients. Figure 3.16 depicts an example of the program. We included the features and a summary of the source code of the Human Arm Simulation in Appendix C.

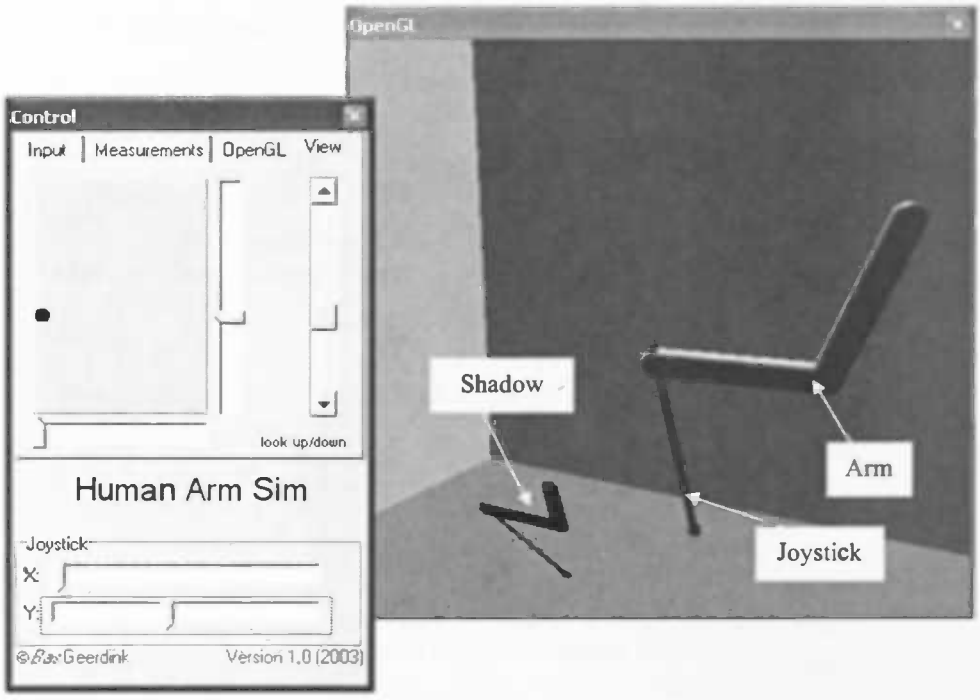


Figure 3.16: A screenshot of the OpenGL dynamic arm simulation. An explanation of the two windows (Control and OpenGL) is in Appendix C.

Chapter 4

Results

After collecting all the data from in total nine test subjects across the two experiments, we analysed all aspects of the computer game and the elbow angles from the goniometer. This chapter will discuss the most important findings. Because of the different nature of the two experiments, we describe them separately.

4.1 Experiment 1

The first experiment, with healthy test subjects, gave us some insights into the behaviour of the children. They all liked the game and kept playing until their time was up. We plotted the stored data variables in graphs. Figure 4.1 and 4.2 show the errors of one test subject in time on position based traject games. Both graphs depict a time period of 300 milliseconds.

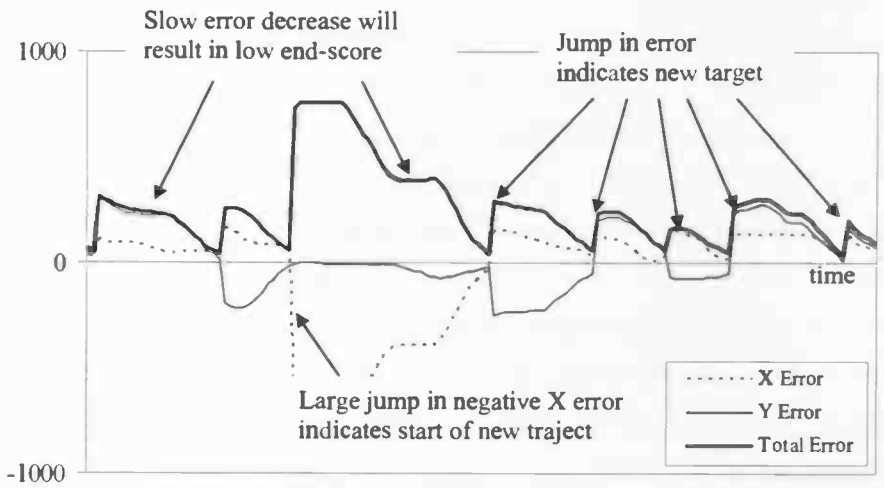


Figure 4.1: An error graph of a test subject at the beginning of the test day.

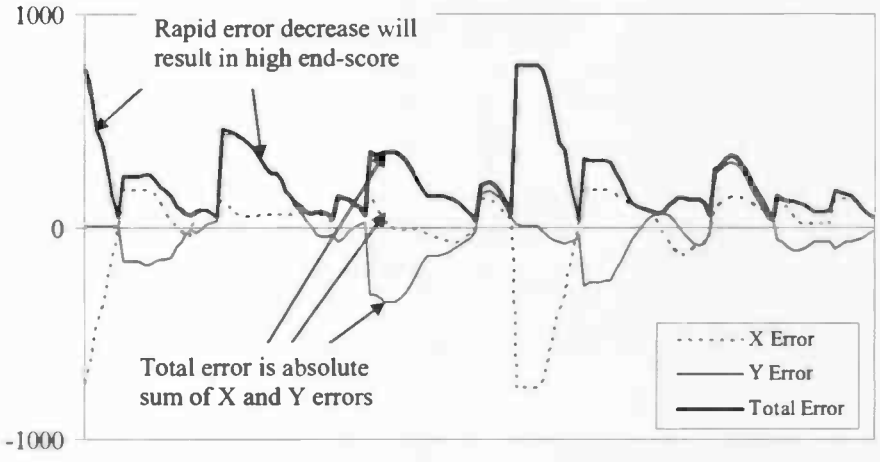


Figure 4.2: Another error graphs of the same test subject. This graph is of the end of the test day. The increase in performance shows from the decrease in time between trials.

Several characteristic features of a traject game become clear from the graphs in figure 4.1 and 4.2. First, the total error jumps from zero to a value below 1000 when a new trial commences. A new trial displayed a new target on the screen, so the test subject had to move the control object towards the target. Doing that, the errors decreased towards zero again, until the total error was sufficiently small and the control object had reached the target. The slope of the total error graph is a good indication of the overall performance of the test subject, as it depicts the speed at which the control object travels towards the target. A steep error slope will result in fast target capturing, resulting in a large cumulative travelled distance across all trials in a test session, because test subjects would then capture more targets within the set two-minute time limit of one game.

The difference between the two movement types (velocity and position based) becomes apparent from figure 4.3 and 4.4. Both graphs show the relation between the joystick and the control object, both in horizontal (X) position. Figure 4.1 shows the velocity based movement type. As the test subject pushes the joystick from left to right, the changes in movement that the control object makes are small. Even when the subject starts to vibrate, the control object on the screen hardly responds. It becomes clear that this movement type results in slower movement and lower end-scores. The position based movement type, on the other hand, results in a direct relation between joystick and control object.

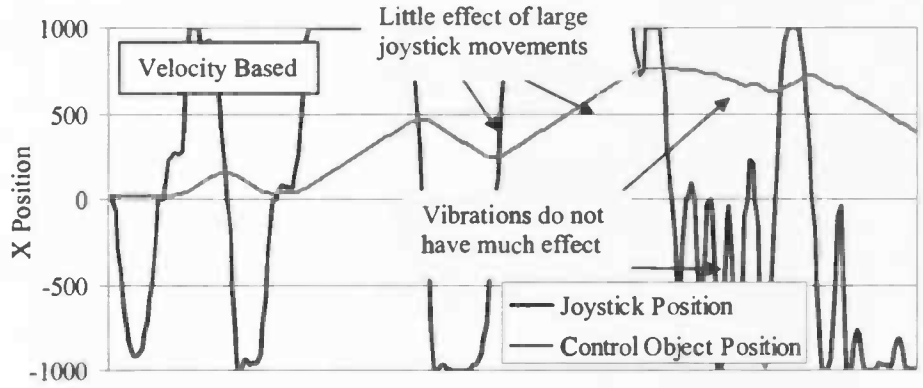


Figure 4.3: A plot of the X position of the joystick and the control object in time during a velocity-based game

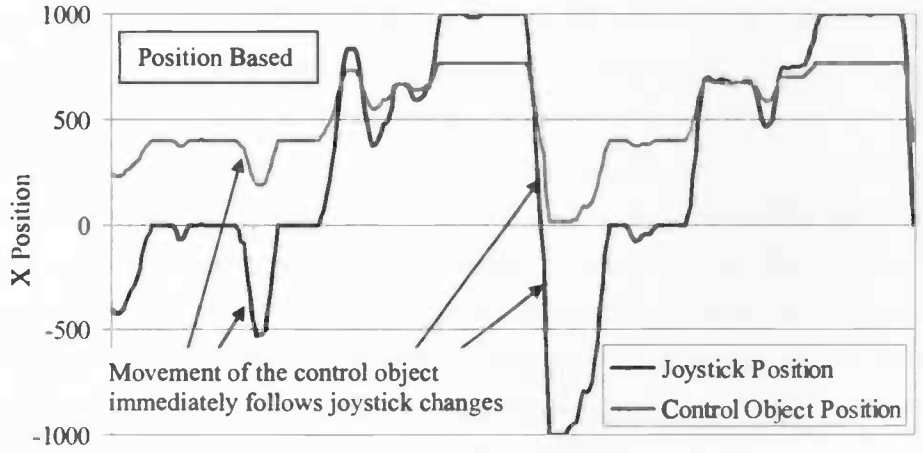


Figure 4.4: A plot of the X position of the joystick and the control object in time during a position-based game

Proof of the children improving their performance in time, indicating a learning trajectory, is in figure 4.5, which displays the average score graph of the test subject on their first and last game sessions. We calculated the average game scores by adding all the relevant games for a point on the graph and dividing that by the number of games added in this way. For example, the average score for all the last traject games of the test subjects was calculated by adding the last traject games of all five test subjects and dividing that by five.

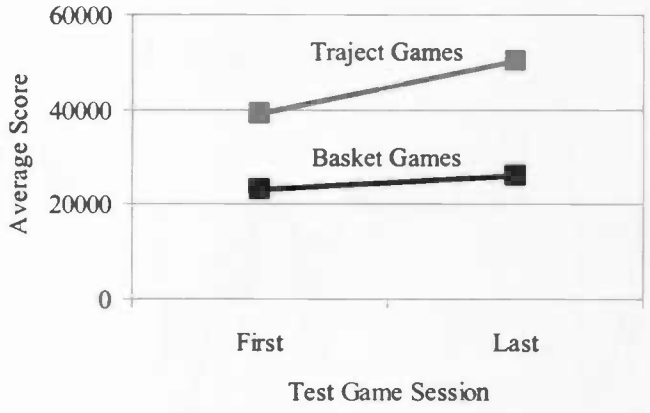


Figure 4.5: Average performance of test subjects across time in experiment 1

As can be seen in figure 4.5, on average all test subjects improved their performance across time. Performance on traject games increased with an average of 29%, while performance on basket games increased with an average of 13%. The differences in the score increase between the game types can be explained by the fact that traject games were played more often, and because they required the subjects to make more arm movements, causing differences to become larger. Several features of the game, such as moving and highlighting targets, did not have a great influence on their performance. We varied those features throughout the test games, but we detected no differences in end-scores when comparing games with several options switched off to games with the options switched on.

4.2 Experiment 2

In the second experiment, the CP children experienced the force feedback assistance as comfortable. Figure 4.6 shows an example of the relation between the error and the force output of the joystick.

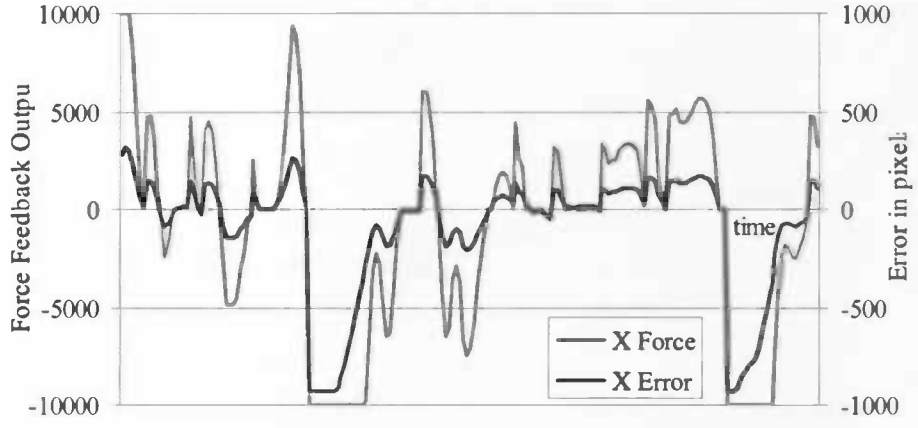


Figure 4.6: An example of the relation between horizontal error and force feedback output. Note that the force output never exceeds 10000 or trails -10000.

Figure 4.7 shows the relation between the force level and the end-scores of the test subjects on their traject games. The force feedback had a positive influence on the performance of the test subjects. The scores are calculations of the total distance between the target objects. Because of the direct relation between joystick position and the position of the control object on the screen, the scores give an accurate measurement of the total amount of movement practice that the children have received. The first (basket-type) game of each test session of five trials was omitted from figure 4.7 because we used those as introduction games to let the children get used to the looks and feeling of the interface here (the 'warm-up' sessions in figure 3.4 and 3.5).

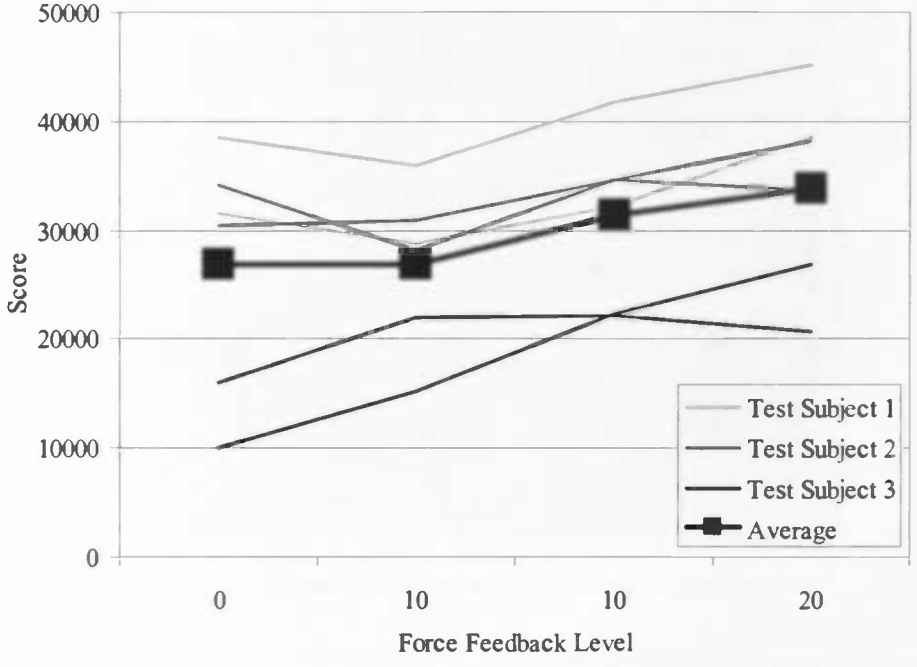


Figure 4.7: A comparison between force feedback settings and resulting end-scores, which represent the total distance that the control object has travelled on screen. Each child played five games on each session, of which the first one is omitted.

The children obtained their best results when the force feedback level was highest. The average performance of the test subjects with high force feedback level was 26% better compared to their performance with no force feedback. There is a correlation of 0.667 between the end-scores and the level of force feedback, which is high enough to conclude that the force feedback has a positive influence.

4.3 Elbow Angles

Physiotherapist that work with CP children will be interested in the actual movement patterns of their test subjects. For that purpose, we recorded the elbow angles through all the test sessions with the goniometer. Output were standard text files, which we plotted into graphs with a data processor (see figure 4.8).

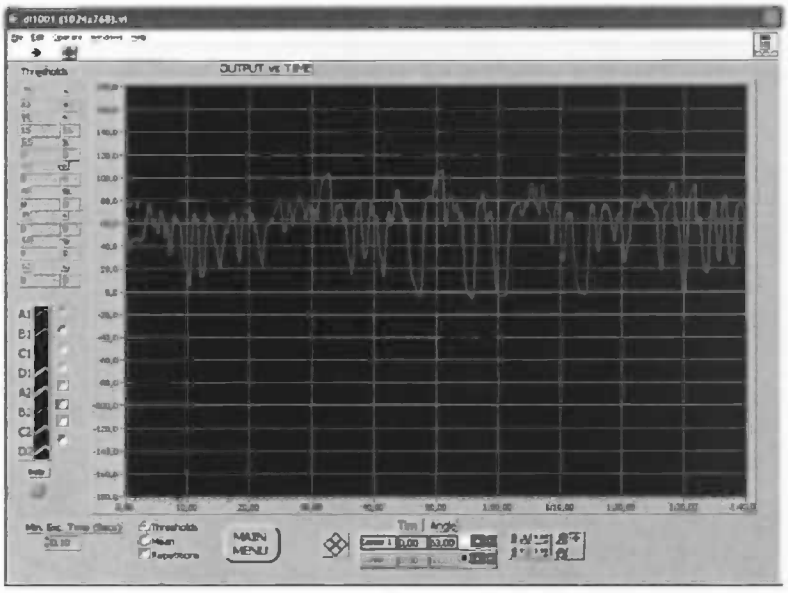


Figure 4.8: A screenshot of a plot from the goniometer data processor

On average, the test subjects with CP moved their elbow joint around an angle of 83 degrees with a standard deviation, which indicates the amount of variation in the elbow movements of the children, of 10 degrees. We analyzed the data files by plotting them into graphs. An example is in figure 4.9. Differences between test subjects' movement behaviour become clear when their goniometer outputs are depicted as such.

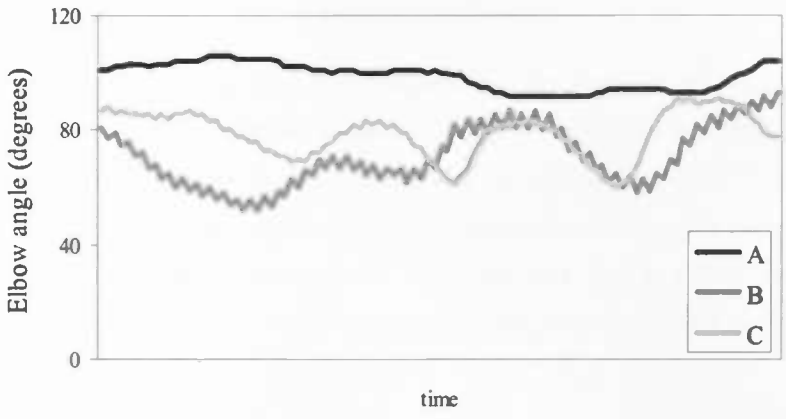


Figure 4.9: Three examples of the angle of elbow joints changing across time. Line A shows a test subject with arm movement problems. Line B shows a vibrating test subject. Line C is the graph of a test subject who performs many movements in time.

In general, the elbow angles will prove helpful for physiotherapists who want to analyze their patients' movement patterns. In addition to that, we searched for a relation between the standard deviations of the elbow angles and the end-scores of the corresponding games. We started by plotting all values in a graph (see figure 4.10).

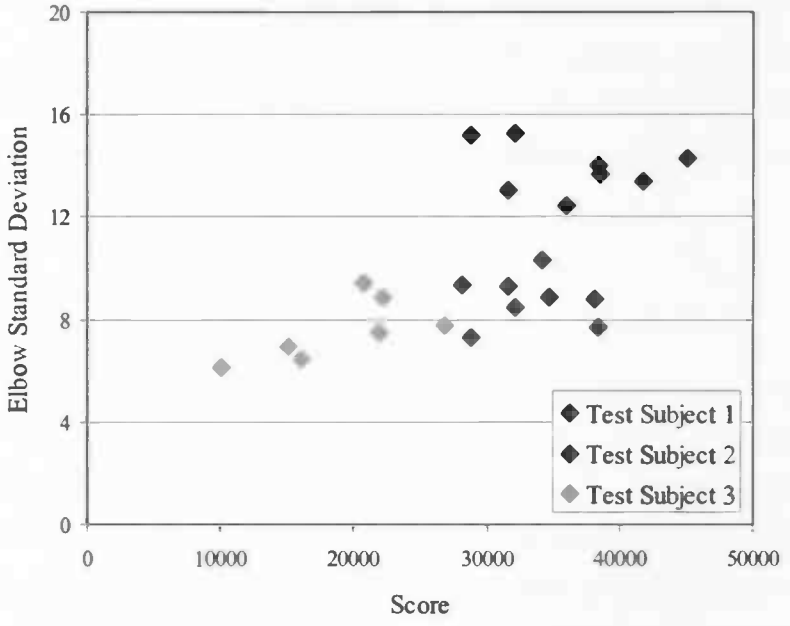


Figure 4.10: The elbow joint standard deviation values of all games, plotted against the corresponding end-scores

When we correlated the end-scores against the standard deviations, we found correlation values of -0,277 for test subject 1, -0,029 for test subject 2, and 0,670 for test subject 3. Therefore, we must conclude that no relation is present between the end-scores of the test subjects and their elbow standard deviations. This means that if test subjects moved their elbows more, they did not get a higher end-score. Therefore, the data from the goniometer can only serve as an evaluation tool.

Chapter 5

Discussion

At the beginning of this project, we knew the problems and imperfections of CP treatment. In our search for improvement of traditional therapy, we contrived the concept of joystick training. Starting with only a few ideas, we worked towards the creation of a computer program, adjustment of the hardware, and a short test period with CP children.

5.1 Summary

The central research question of the project was: will the rehabilitation of children with CP benefit from short exercises with a force feedback joystick interface? To reach a conclusion about that question, we set up two experiments. In the first experiment, we tested five healthy test subjects with a computer game that was controllable with a joystick. We used an extended joystick in the first experiment but did not exert any force feedback output. In the second experiment, we tested three children with CP. They took place in an ergonomic adjustable chair and used a joystick that exerted force feedback. Furthermore, a goniometer was attached to the test subjects, which measured the elbows angles during the test sessions.

In both experiments, we collected a large amount of data from each game. Static data such as the initial game settings and the end-scores, as well as dynamic data such as the position of the control object and joystick coordinates, were recorded for each test subject. With these data, we tried to get a clear view of the children's movement behaviour throughout the test sessions. Plotting the various variables into graphs provided insights about the test subjects, such as their movement speed, arm vibrations, and the amount of elbow joint flexing.

In this scope of monitoring the test subjects, we created the Human Arm Simulation, which is a dynamic software tool for playing back the test sessions of the children in a virtual three-dimensional space on the computer screen. The output of

this tool should be beneficial for physiotherapists who want to evaluate their patients in a time space of a few months.

5.2 Conclusion

The results presented in chapter 4 show that force feedback had a positive influence on the performance of children suffering from CP; the more force was applied, the better the children performed. Effects of fatigue could be the reason that the scores of test subjects 1 and 2 decreased after three trials, but the tests of subject 3 show that in the ten minutes that the children played the game, fatigue had no influence on the score. Therefore, the improvement in end-scores of the test subjects can be largely assigned to force output. In other words, a big factor for determining the movement speed of the children was the level of force feedback output.

There was no significant relation between the elbow angles and the performance of the children, which means that test subjects could play the game either with or without flexing their arms. In the latter case, the children would tend to bend over, controlling the joystick with their upper body more than with arm movements. This is an unwelcome effect, since the purpose of the game was to let the children practise their arm movements. Prevention of this effect should begin by instructing the children to move their body as little as possible and to concentrate on moving their arms. Another option was to force the children's shoulders into a set position, so that their upper bodies cannot move, for example by tying them to the chair. This might solve the problem but it would not be comfortable for the children. Since on of the goals of the research project was to let children exercise in a pleasant way at home, it will be clear that we have not considered utilizing this option.

Although it is difficult, if not impossible, to reach a good conclusion about a research project with as few as three test subjects with CP, at this point, we take the liberty to conclude that force feedback therapy is a possible way of treating CP children attractively and efficiently. The advantages of this type of treatment include the possibility for the children to work at home, the excellent evaluation options thanks to the goniometer and the Human Arm Simulation, and the fun of playing a computer game.

5.3 Future Research

Since medics still know little about the causes, symptoms, and effective treatment methods of CP, research after this disease should continue in the first place. We suggest filling in part of the treatment of CP with robotic therapy such as force feedback assistance. Fortunately, people across the world invest a large amount of time and money into projects that search for better ways of treating CP patients.

The aim of future research should be at using the joystick in a home environment. Our prediction is that children will benefit greatly from practising with the software each day for about fifteen minutes. After a while, a supervisor can change the software settings to make the exercises harder, for example by applying less force feedback or even setting the force in a negative direction so the children have to push against the joystick to reach a target. Some aspects come to mind when considering home exercises for children. For example, patients' supervisors have to make some environmental requirements, such as the availability of a personal computer and an extended force feedback joystick. Initially, hospitals that own joysticks could loan them out to for families with a CP child. Since a personal computer is available in most households nowadays, connecting the joystick to it, and assuring that the child is in a comfortable playing position, would be enough to offer daily exercises in a home environment. Eventually, patients could send their static and dynamic game data via e-mail to a physiotherapist who can analyze them with assistance of the Human Arm Simulator.

Appendix A

Implementation of Game 1

In the first game, used to test healthy test subjects, there were three game screens: a login screen, an options screen, and the main game screen. When the game was started, the login screen (see figure A.1) popped up. Test subjects could select a name or enter a new name, and press the 'OK' button.

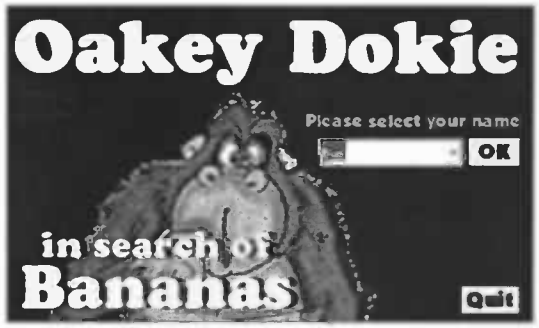


Figure A.1: The login screen of 'Oakey Dokie'. Users could select their name from a drop-down menu, or enter a new name in the text field.

The next screen showed the configuration options (see figure A.2). In this window, the user could alter all game options. It showed the highscore as well as the total number of games that the test subject has played. The 'Reset' button was enabled if the user had played at least one game before. In that case, his old settings were stored in a text file (see figure A.3). When the user pressed the 'Reset' button, that file was read and the game placed all settings back to their old values. 'Start' launched a new game with the current settings, and 'Quit' exited the game. The checkbox that reads 'Force Feedback' was not enabled because we did not yet implement force feedback in this version of the game.

Figure A.2: The options screen. Users could choose between two game types and two movement control types. Three difficulty options were selectable and there was a possibility to set a time interval within which the test subject had to capture the banana.

```
1     Bas
2     F T T F T X B 6985 /2\ [12]
3     Carrie
4     F T T F T X T 4218 /11\ [6]
5     Max
6     F T F T F X B 2579 /9\ [8]
7     Natalie
8     F T T F T X T 5376 /2\ [10]
9     Sarah
10    T T F T T X T 6324 /5\ [6]
```

Figure A.3: An example of 'users.txt'. The odd line numbers contain the names of the users, the even line numbers contain values for all variables of the options screen, including highscore, number of games played, and force output level, separated by spaces.

The game, called 'Oakey Dokie' had two different game types: *basket* and *traject*. In a basket game (see figure A.4), the monkey started in the centre of the screen. A banana appeared at a random position. The user could control the monkey by pushing the joystick. The goal of the basket game was to let the monkey grab a banana. Once he had grabbed it, a basket appeared in the centre of the screen. The next task of the user was to make the monkey dump the banana in the basket. Therefore, the movements of the basket games were a kind of to-and-fro motion from the centre to a banana and back.

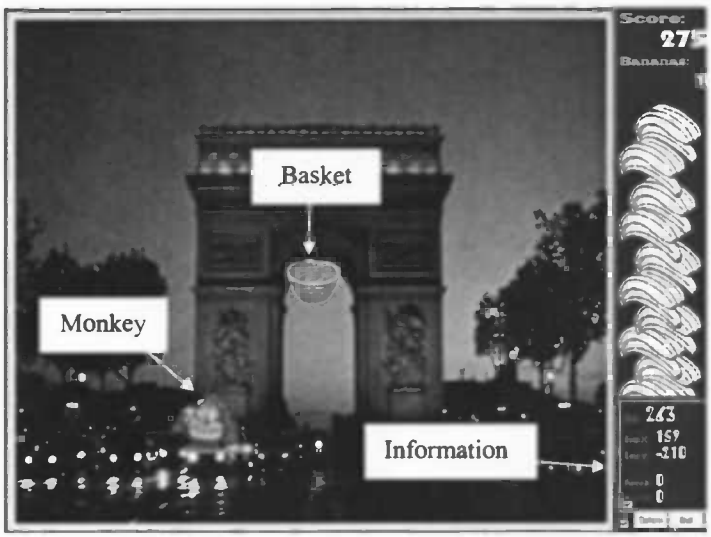


Figure A.4: A screenshot of a basket game. The monkey had to capture one banana, which appeared at a random position, and bring it to the basket in the centre of the screen. The information panel on the bottom-right could be activated by clicking a checkbox.

In a *traject* game (see figure A.5), the monkey started at blue dot towards the left of the screen. It had to capture five bananas, and then reach the basket on the right side of the screen. Hence, the test subject had to reach seven targets in total on the screen. They were connected with a yellow line. When the test subject had directed the monkey to the final target, the basket, a new level started, in which the first target was again the blue spot. The horizontal (X) positions of the targets were set, where the software randomized the vertical (Y) positions of the five bananas at the beginning of each level. This type of game required the user to make a motion that was mostly from left to right. More information about the game types is in section 3.2.3.

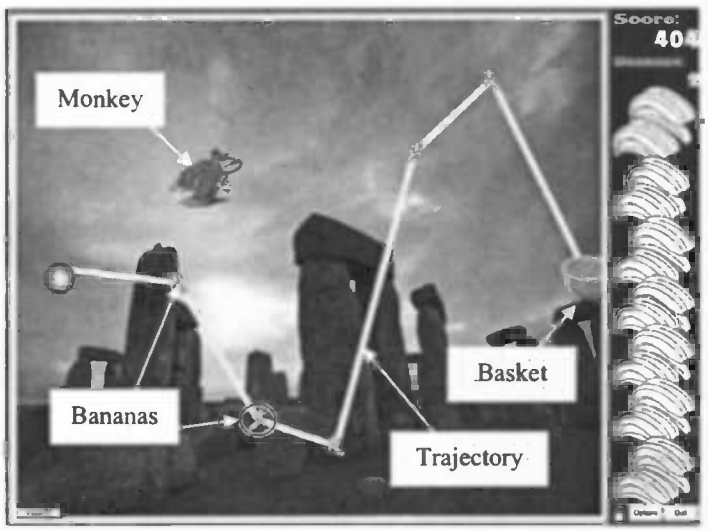


Figure A.5: A screenshot of a traject game. The monkey started at the blue spot and had to capture five bananas along the yellow line and dump them in the basket.

Besides the game type, the setting that had the most influence on the game is the movement type, which regulated the relation between the position of the joystick and the control object on the screen. On the options screen, the test subjects could choose between 'Velocity Based' and 'X/Y Based'. When the velocity-based movement type was selected, the monkey accelerated in the direction that the joystick pointed. His speed was dependant, through a linear function, on the angle that the joystick handle made. Therefore, if the user pushed the joystick a little to the left, the monkey would slowly walk towards the left. If the user pushed the joystick handle towards its extreme right-forward position, the monkey would move fast towards the upper-right.

The position-based ('X/Y Based') movement type resulted in a direct relation between the joystick position and the position of the control object on the screen; when the test subjects pushed the joystick towards its extremities, the monkey would instantly walk towards the corresponding position. More information about the movement types is in section 3.2.2.

The remainder of this appendix contains a summary of the most important procedures (■) and functions (□) of Game 1. **Formcreate**, **Init**, and **CreateDataFiles** were only used once for each game. Procedure **Timer1Timer** was called every fifty milliseconds, and regulated the other procedures and functions. The formula of Pythagoras (see equation 3.8) was used frequently.

■ **FormCreate**

- initialize the game screen on resolution 1024 x 768;
- create *TempImage*, which is a virtual image in the memory of the computer. The actual image that is displayed on the screen is a copy of *TempImage*;
- create the five images called *Bananas*;
- create and fill 24 arrays with images for the animation of the monkey. All images are bitmaps from the directory 'pics'.

■ **Init**

- call **CreateDataFiles**;
- set position of monkey to the center or the left of the game, dependant on the game type;
- calculate the end-time;
- write line in file *EventData* with information about the game that just started, such as: duration, game type, time interval and movement type of the joystick;
  - call **PlaceBanana** which places bananas at random positions on the screen;
  - calculate Dist, which is the total distance between the monkey and banana in pixels, by calling function CalcDist.

■ **CreateDataFiles(*Number* : Integer, *Name* : String)**

- create a directory on the hard disk for the user if it doesn't already exist;
- create numbered files *EventData* and *XYData* in the directory which will store all the recorded data.

■ **PlaceBanana**

- assign a random value to *BananaDir*, which is used if the option 'Moving Bananas' on the options screen is checked;
- in a basket game: place *ImageBanana* on random position on the screen;
- in a traject game: place *Bananas[1..5]* on random y-coordinates on the screen.

☐ CalcDist : Integer

- calculate *x* and *y*, which are the errors between the monkey and current target banana;
- return an integer value indicating the absolute error in pixels by using the formula of Pythagoras.

■ Timer1Timer

- increase the value of *Time* by 1. Timer1Timer is called every 50 milliseconds;
- update the DirectX input component, which retrieves the joystick position;
  - call **MoveBanana** if the option is checked;
  - call **CalcCircle** if the option is checked;
- calculate *TempDist* by calling **CalcDist**, which also updates the errors *x* and *y*;
  - call **CalcForce** if the option is checked, which it never was in version 1 of the game;
- calculate the position of the monkey on the screen, dependant on the movement type;
  - call **AnimMonkey** if *Time* is odd, so every 100 ms;
  - call **DrawImage**;
- check if the monkey stays within the screen;
- write a line to *XYData* with the joystick position, the position of *ImageMonkey*, and the errors *x*, *y*, and *TempDist*;
  - call **Grab** if *TempDist* has a value lower than 60 pixels;
- check if the endtime has been reached and close the game if it has;
- check if the highscore has been improved.

■ MoveBanana

- in a basket game: move *ImageBanana* to the direction indicated by *BananaDir*;
- in the traject-game: move *Bananas[Trial]* up or down;
- check if the moving banana stays within the screen.

- **CalcCircle**
  - calculate the postion of the circle around *ImageBanana*, *ImageBasket*, *ImageStart*, or *Bananas[Trial]*. The circle indicates the next target for the monkey.

- **CalcForce**
  - assign values to *TempPowerX* and *TempPowerY*, which are calculated with linear functions of x and y;
  - start the force feedback effects in the list, dependant on the errors x and y.

- **AnimMonkey**
  - temporary variable *we* become 4 if *HasBanana* is true, and 0 otherwise. That is to discriminate the images in the arrays: 0 to 3 displays a normal monkey, 4 to 7 is a monkey with a banana in his paws;
  - dependant on the position of the monkey and the previous postion of the monkey, *ImageMonkey* becomes *MoveLeft[AnimMonkey + we]* or *MoveRight[...]* or *Still[...]*. In that way, the animation shows by cycling through the arrays of monkey frames;
  - increase the integer value of *AnimMonkey* by 1 or make it 0 if it is larger than 3.

- **DrawImage**
  - clear the canvas of *TempImage*;
  - in *TempImage*, draw the following objects in this order:
    - the background image (stretched);
    - the traject lines if the option is checked on the options screen;
    - *ImageBasket* in a traject game or if *HasBanana* is true;
    - *ImageStart* in a traject game;
    - *ImageBanana* in the basket game if *HasBanana* is false, or *Bananas[1..5]* in a traject game;
    - *ImageMonkey*;

- *ImageCircle* if the highlight-option is checked on the options screen.

- copy the contents of *TempImage* to the screen.

■ **Grab**

- increase the value of *Trial* by 1;
- update the score label;
  - call **PlaceBanana** in the basket-game;
  - assign the return-value of **CalcDist** to *Dist*;
- write a line to *EventData* in which the time and score are recorded;
  - call **IncLevel** if the next level is reached. In a basket game the level distribution is a linear function of *Trial*, in a traject game the user goes to a next level if the value of *Trial* is larger than 6;
  - call **PlaceBanana** in a traject game if the value of *Trial* is larger than 6;
- assign a random value to *BananaDir* in a traject game.

■ **IncLevel**

- view the next image of bananas on the right side of the screen to show the level build-up;
- load a random image from directory 'pics' to the background-image.

Appendix B

Implementation of Game 2

Since Game 2 is an extension of Game 1, we will only list the changes we made along the implementation path. In experiment 2, we tested children with CP. The software had to contain force feedback output. We decided to make two different games to make the game more attractive for children. We designed a football game, which only differed in appearance from the monkey game. Where the monkey game was about collecting bananas, the user of the football game had to manoeuvre a player across a football pitch. The basket game was replaced by 'dribbling practise' and the traject game was named 'passing practise'. In the first screen (see figure B.1), users could choose between the two games, 'Oakey Dokie' and 'Force Football'. Next to that, the login procedure was not altered in comparison to Game 1.

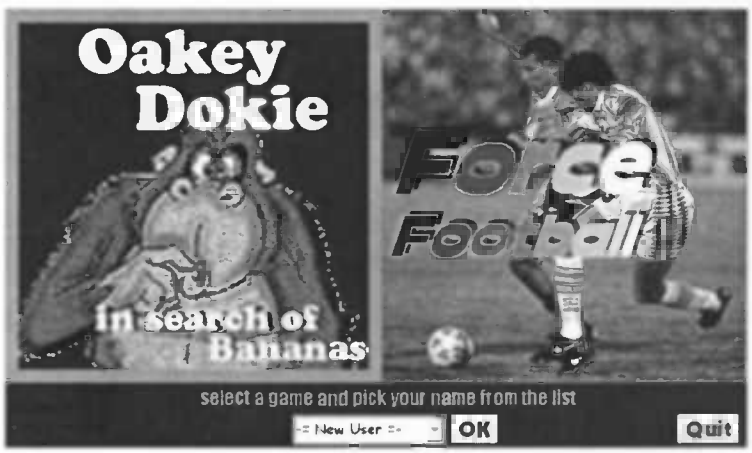
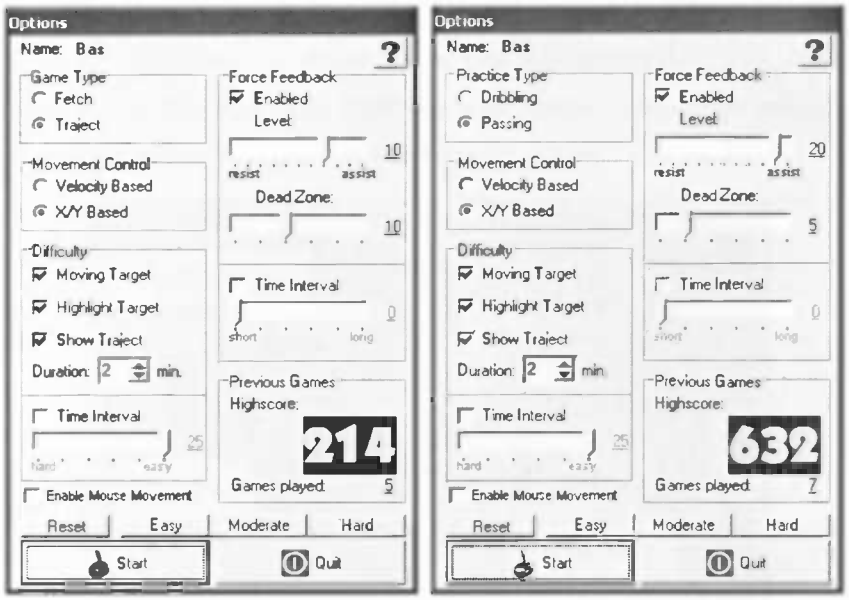


Figure B.1: The login menu of game 2. Users had to choose between 'Oakey Dokie' and 'Force Football' by clicking on the pictures. After selecting a name from the drop-down menu or entering a new name the options screen appeared.

When the user had pressed the 'OK' button, the options screen appeared (see figure B.2). New options in comparison with Game 1 were the force feedback settings. There were three force feedback settings, which were enabled if the checkbox was checked. The 'Level' bar ranged from -25 to 25 and set the intensity of the force feedback (see section 3.2.4). The 'Dead Zone' bar ranged from 0 to 25

and set the area around the target in which there was no force feedback. The 'Time Interval' set a period after which the force feedback would take place. We did not use this setting in our experiments, but it could be helpful in future experiments.



(A) (B)

Figure B.2: Screenshots of the options panel of (A) 'Oakey Dokie' and (B) 'Force Football'

As can be seen in figure B.2, we created three buttons with standardized settings: easy, moderate, and hard. Pressing one of these buttons caused all settings to take certain values, which are displayed in figure B.3. All three buttons caused the game type to switch to *traject* and the movement type to switch to *position based* (see section 3.2.2 and 3.2.3)

				Force Feedback		
Setting	*Moving Target*	*Highlight Target*	*Show Traject*	*Enabled*	*Level*	*Dead Zone*
Easy	yes	yes	yes	yes	20	5
Moderate	no	yes	yes	yes	10	10
Hard	no	yes	no	no	-	-

Figure B.3: The three possible standardized settings of game 2

We added a button with a question mark on it for bringing op the game manual, which is included in full in Appendix C. Finally, we created a checkbox labelled 'Enable Mouse Movement'. When checked, playing the game was possible with the

computer mouse by moving it around the screen. We added this option to show the game to people who do not own a joystick.

When the test subject had adjusted all settings to his liking, the start button engaged a new game. Figure B.4 and B.5 depict the two main screens of the games. In comparison with Game 1, the action screen had increased in size, there were indicators for standard and force feedback time interval when those settings were enabled, and the background pictures had been simplified.

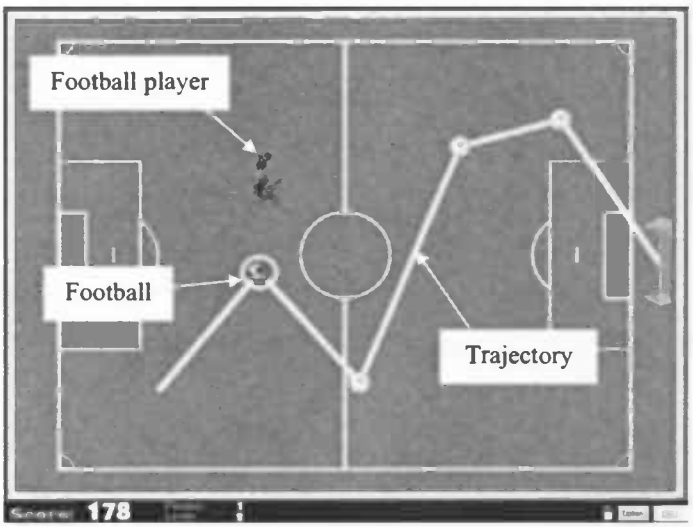


Figure B.4: A screenshot of 'Oakey Dokie'

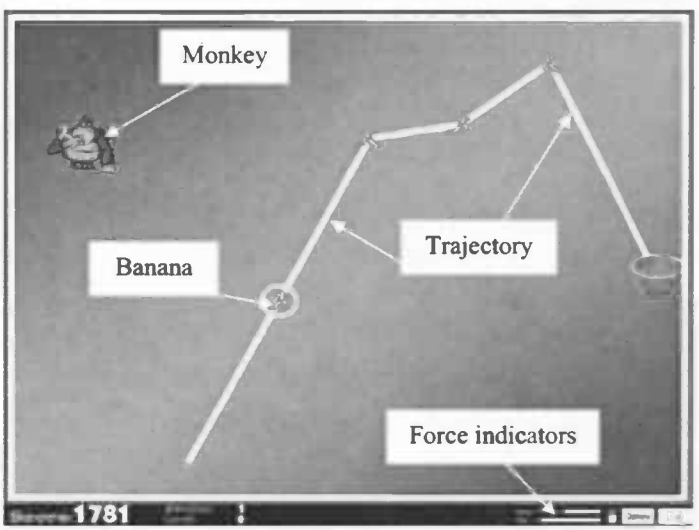


Figure B.5: A screenshot of 'Force Football'

Appendix C

Implementation of Arm Simulation

We constructed the 'Human Arm Simulation', which was based on the OpenGL programming technique, to allow physiotherapists to play back their patients' arm movement training and evaluate their progress. In this appendix, we summarize this software tool. The program consisted of two windows, *Control* and *OpenGL*. In the control window, the lower part showed the horizontal (X) and vertical (Y) position variables of the joystick as trackbars. The trackbars could be adjusted by dragging the indicators with the computer mouse. The upper part of the control window contained four screens to manipulate the program, which are depicted in figure C.1.

(A) (B) (C) (D)

Figure C.1: The four screens of the Control window: Input (A), Measurements (B), OpenGL (C), and View (D)

We will discuss each of these screens of the control window. First, in the *Input* screen, the user could choose three input options for the program. The first of those options was 'Manual', in which the user had to change the virtual X and Y joystick positions by dragging the corresponding trackbar indicators by using the mouse. The second input option was 'Joystick', which was selectable if a joystick was connected to the computer. The program read the joystick position and set the X and Y trackbars to those values. The third input option was 'File', which was the most interesting as it brought up a screen where the user could select a text file, which

would be used as the sampling file. Every fifty milliseconds the program read a line from the input file and extracted the X and Y positions of the joystick if the layout of the file was correct. The program could only read text files that were created with the second version of the joystick game. They had names such as 'XYData(username)1.txt' and were stored in the directory 'Data/(username)' (see section 3.5.2 for an overview of XYData text files).

In the *Measurements* screen, the user had to enter eight values, which determined the place of the joystick and the shoulder of the test subject. Values such as the arm length and the distance between the joystick base and the floor set the initial position of the joystick-arm set-up in the virtual room. Additionally, a line labelled 'Reach' showed the distance in space between the hand and shoulder of the subject.

In the *OpenGL* screen of the control window, three settings determined the view of the OpenGL window. Test subjects could enable or disable light effects and shadows, and the 'Graphics Level', which determined the detail level of the OpenGL simulation, could be set from zero to one hundred, in steps of one. The standard settings were enabled light effects and shadows, and a graphics level of thirty.

Finally, in the *View* screen, the user could change his perspective on the joystick situation by virtually walking across the room and looking up- or downwards. There were three trackbars for this purpose: the X and Y position of the viewpoint, and the viewing direction in the Z direction.

Next to the control window, the OpenGL window showed the result of the settings and the joystick coordinates. Figure C.2 shows three examples of the dynamic simulation.

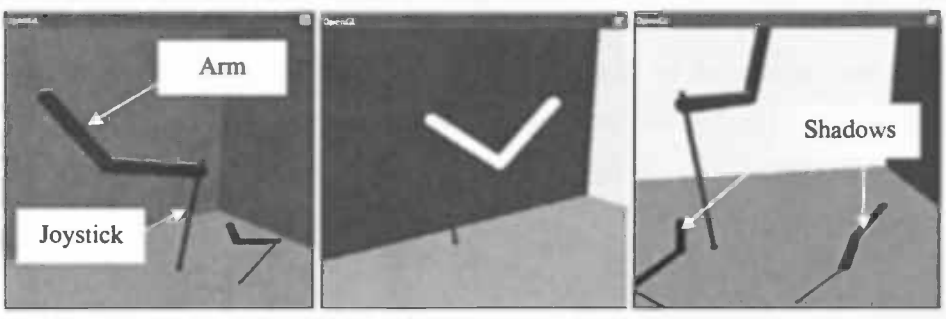


Figure C.2: Three screenshots of the OpenGL window

To realize a three-dimensional model of the shoulder-elbow-hand-joystick system, we sketched all points in space into the image of figure C.3.

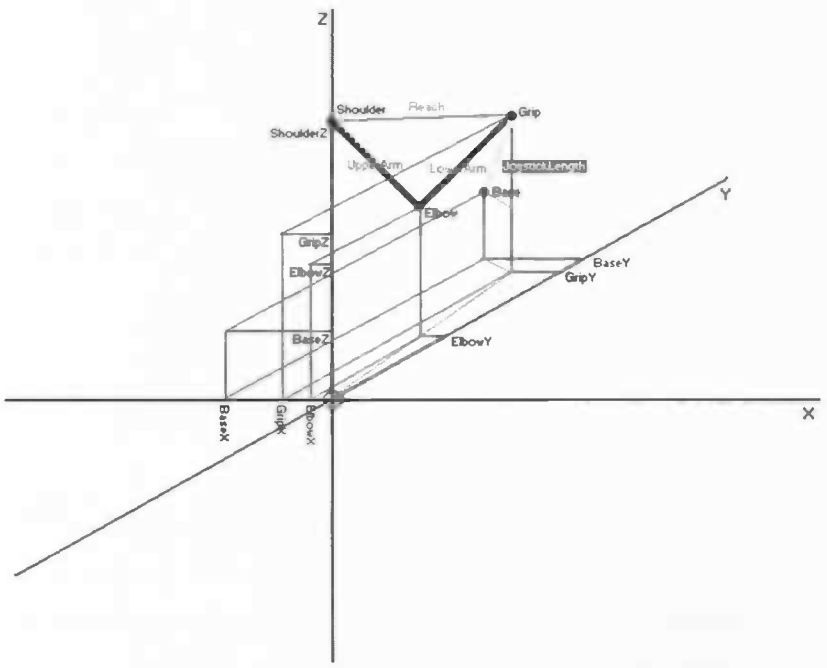


Figure C.3: Situation analysis of the shoulder-elbow-grip-base coordinates

After the design on paper, implementation followed. We used the following goniometric equations in the source code:

$$\alpha(\text{radians}) = \pi/180 * \alpha(\text{degrees}) \qquad \text{(C.1)}$$

$$\cos(\alpha) = (-A^2 + B^2 + C^2) / (2 * B * C) \qquad \text{(C.2)}$$

The remaining part of this appendix is a summary of the source code of the Human Arm Simulation. All procedures (■) and functions (□) are listed, and were processed in the same manner as with both joystick games.

■ **FormCreate**

- create and activate OpenGL window;
- create templates for cylinders and spheres;
  - call **GLInit**;

- call ArmInit.

■ **GLInit**

- set viewing direction and mode of the OpenGL matrix;
- set background to black color;
- enable lighting;
- create a virtual light bulb at a position on the ceiling which radiates light in a set direction;
- create a diffuse light source for light in the whole virtual OpenGL space;
- set properties for the materials in the OpenGL space, such as shininess and the amount of light that is absorbed.

■ **ArmInit**

- extract the length of the upper- and forearm and the length of the joystick from the *Measurements* tab of the Control window;
- determine the position of the base of the joystick and the shoulder of the test subject using values from the *Measurements* tab. We represented each coordinate in the OpenGL space by an array of three variables, 0 to 2. For example, the shoulder is represented as *Shoulder[0..2]*.

■ **Timer1Timer**

- this procedure is called every 50 milliseconds;
- if a text file is loaded as input for the simulation, a line is read from the input file and the X and Y positions of the joystick are extracted from it;
- if the input is set to 'Joystick', the X and Y positions are calculated;
- call **CalcGrip**;
- call **CalcElbow**;
- call Draw.

- **CalcGrip**
  - calculate the angles of the joystick in X and Z directions and convert them from degrees to radians;
  - calculate the X and Z coordinates of the joystick base (*Grip[0]* and *Grip[2]*) using the sinus of the joystick angle;
  - calculate the Y coordinate of the joystick base (*Grip[1]*) using the formula of Pythagoras. Now, the points *Base[0..2]*, *Shoulder[0..2]*, and *Grip[0..2]* are known.

- **CalcElbow**
  - calculate the distance from the shoulder to the grip point of the test subject (Reach) in three dimensional space using the formula of Pythagoras twice;
  - calculate *Alpha*, which is the angle between shoulder and grip using the Arctangent formula;
  - calculate *x*, which is the angle grip-shoulder-elbow, using the Cosinus formula;
  - calculate *Beta*, which is the actual angle grip-shoulder-elbow, using the Arccosinus formula;
  - calculate the X, Y and Z coordinates of the elbow joint (*Elbow[0..2]*).

- **Draw**
  - determine the virtual position of the view in the OpenGL space and the direction to look at by extracting the values from the *View* tab;
  - draw the floor of the OpenGL room by calling the OpenGL procedure **glVertex3f** four times: once for each corner of a plane in three-dimensional space. Do the same for the ceiling and walls;
  - call **DrawArm**;
  - if the 'Shadow' option on the *OpenGL* tab is enabled, draw the shadows of the arm and joystick on the floor by calling **DrawShadow** four times: once for each shadow.

- **DrawArm**
  - draw four spheres from the template, at the positions of the base, shoulder, grip, and elbow;
  - call **DrawUpperArm**;
  - call DrawLowerArm.

- **DrawUpperArm**
  - calculate *OE*, which is the distance between the origin *O[0,0,0]* and *Elbow*, by using the formula of Pythagoras;
  - calculate *Alpha*, which is the angle over which the upper arm is rotated in the X/Y plane, by using an ArcTangent formula;
  - calculate *Beta*, which is the angle over which the upper arm is rotated in the plane *O-Elbow-Shoulder*, by using an ArcTangent formula;
  - draw a cylinder from the template with length and direction of the upper arm, by translating and rotating the OpenGL projection matrix.

- **DrawLowerArm**
  - calculate *EG*, which is the distance between *Grip* and *Elbow*, by using the formula of Pythagoras;
  - calculate *Alpha*, which is the angle over which the forearm is rotated in the X/Y plane, by using an ArcTangent formula;
  - calculate *Beta*, which is the angle over which the forearm is rotated in the plane *O-Elbow-Shoulder*, by using an ArcTangent formula;
  - draw a cylinder from the template with length and direction of the forearm, by translating and rotating the OpenGL projection matrix.

- **DrawShadow(*P* : LightPosition)**
  - calculate a matrix ShadowMat by calling ShadowMatrix(Ground, P). Now, the area on which everything is drawn is the ground floor;
  - draw four spheres from the template, at the positions of the base, shoulder, grip, and elbow;
  - call **DrawUpperArm**;

- call DrawLowerArm.

☐ ShadowMatrix(*V* : Area, *P* : LightPosition) : Matrix

- calculate Point, which is the normal vector of area V, by calling GetNormal(V);
- calculate and return *ShadowMatrix[0..3][0..3]*, based on *Point* and *P*.

☐ GetNormal(*V* : Area) : Point

- calculate point *Erg[0..2]*, which is the direction of the normal vector of area *V*;
- calculate and return GetNormal, which is the normal vector of area V, by calling NormalCut(Erg).

☐ NormalCut(*P* : Point) : Point

- calculate the length of normal vector *P* by using the formula of Pythagoras;
- calculate and return *NormalCut[0..2]*, which is the normal vector with length 1, by dividing *P[0..2]* by the length of the original normal vector.

Appendix D

Game Manual

Because the actual goal of the project was to let children work with the joystick and software in their own homes, we wrote the following manual for the game.

```
+-----------------------------+
| Oakey Dokie / Force Football |
|         Joystick Game        |
+-----------------------------+

  ---==== Version 2.1 ====---
```



```
Last update: 16/10/2003
=====================================================================

1) INTRODUCTION
---------------------------------------------------------------------
Welcome. This computer game was developed in 2003 with the
purpose of aiding disabled children. The game can be played with
any joystick that is connected to a computer, but a Force
Feedback joystick is preferred.

There are two games in this software package. The nature of the
games is the same, the difference lies in the graphical
interface. In this manual all features of the games will be
explained. Please do not hesitate to contact me if you have any
questions regarding the game after reading it. Enjoy the game!

2) SYSTEM REQUIREMENTS
---------------------------------------------------------------------
Minimum configuration:
 * Pentium 133 or higher
 * 64MB RAM
```

* 5MB free harddisk space
* 800x600 screen resolution
* Microsoft Windows 95/98/ME/NT/2000/XP
* Mouse
* Joystick

Preferred configuration:
* Pentium II or higher
* 128MB RAM
* 5MB free harddisk space
* 1024x768 screen resolution
* Microsoft Windows 95/98/ME/NT/2000/XP
* Mouse
* Force Feedback Joystick (extended)

3) FILE LIST

The game requires the following files in order to run properly.

In main directory:

* manual.txt	- this manual
* program.exe	- the executable game file
* users.txt	- a list of people who played

In directory 'pics':

* ball[0-5].bmp	- frames of a rotating football
* ball.bmp	- a light colored ball
* banana[0-1].bmp	- two bananas
* basket.bmp	- a basket
* bg[0-8].bmp	- nine simple backgrounds
* circle[1-13].bmp	- frames of a highlighting circle
* frame[0-9].bmp	- frames of a highlighting square
* goal[0-1].bmp	- two goals
* monkey-b-mL[0-3].bmp	- frames of monkey walking left with a banana
* monkey-b-mR[0-3].bmp	- frames of monkey walking right with a banana
* monkey-b-s[0-1].bmp	- frames of monkey standing still with a banana
* monkey-mL[0-3].bmp	- frames of monkey walking left
* monkey-mR[0-3].bmp	- frames of monkey walking right

* monkey-s[0-1].bmp - frames of monkey standing still
* pitch.bmp - background of football game
* player-b-mL[0-15].bmp - frames of football player walking left with ball
* player-b-mR[0-15].bmp - frames of football player walking right with ball
* player-b-s[0-1].bmp - frames of football player standing still with ball
* player-mL[0-15].bmp - frames of football player walking left
* player-mR[0-15].bmp - frames of football player walking right
* player-s[0-1].bmp - frames of football player standing still

The variable pictures for the frames and background enable you to insert your own picture into the game. Do not experiment without making a back-up of the original pictures!

There is also a directory 'data' in which all game information is being stored. The files in it are not necessary for played the game.

4) STARTING A GAME

When the executable file is launched, the start menu appears. The two sides of the screen show the games you can choose from: 'Oakey Dokie (in search of bananas)' or 'Force Football'. Make a choice by clicking with your mouse on one of the pictures. There is an list at the bottom of the screen which contains the names of all people who have played the game, loaded from the file 'users.txt'. If you have played before, click on your name. If you are new to the game, select 'New User'. Next, click on the 'OK' button to continue. Your option settings will be loaded and the next screen to appear is the options screen (see chapter 8) You can always end the game any time by clicking on the 'Quit' button.

5) MAIN SCREEN

After clicking the 'Start' button on the options screen, the main

game screen appears. The biggest part is the actual game area, where you'll see Oakey Dokie or the football player executing your commands. At the bottom of the screen is some information available: the score you have achieved so far, the number of bananas/balls you have captured in this level, and the number of the level you're playing in. The level will increase if you have captured ten bananas in the basket, scored ten goals, or completed the traject in the 'Traject Game' or 'Passing Practice'. If you enabled one of the time intervals on the options screen, the interval will be displayed by a green or red bar that represents the amount of time you have before the force feedback begins or the remaining time for the trial. Finally, the button 'Options' can be clicked to bring up the options screen (your present game will be ended) and the 'Quit' button will exit the game.

6) OAKEY DOKIE

The monkey you saw on the start screen is Oakey Dokie. As most monkeys he likes bananas. In this game you can help him find as many bananas as possible! On your computer screen you'll see Oakey and a banana he can grab. But since he is a little lazy, he doesn't walk to the banana himself. You'll have to steer him in the right direction! So, if you move your joystick you'll see that Oakey walks in the direction you 'tell' him to.

6.1) Fetch game

If you do well, Oakey will grab the banana. You can see he holds it in his hand. You also see a basket appearing in the centre of the screen. Try to move Oakey towards the basket. If you're close, he will drop the banana in the basket and keep it. When the game is finished, he'll start eating all the bananas in the basket you collected for him!

6.2) Traject game

Now, because this monkey is pretty hungry he has to grab a many bananas as possible. He may not look so smart, but he figured out that he can hold on to five bananas at a time. So now you'll have to make Oakey grab five bananas that are scattered across the screen. When he has grabbed all five, he has to dump them in the basket again! You'll notice that this goes much faster than the

fetch technique.

7) FORCE FOOTBALL

This game is about playing football: in other words, scoring goals! The main character is a young football player who is at the beginning of his career. Since he wants to be a world class star when he is older, he has to practise a lot. You can help him do that. The player is not interested in defending, he just wants to score a lot of goals!

7.1) Dribbling practise

To become a good player, there has to be some dribbling practise. When you begin playing, our future football hero is in the centre circle of the pitch. You'll also see a rotating ball on your screen. You can control the player by pushing the joystick in the direction you want him to go. Try to get the ball! If you have reached it, you'll see that the player dribbles with the ball now. Also notice the goal appearing on the pitch, that is where you can score. So now make him run to the goal, score, and start your next practise run!

7.2) Passing practise

Good football players will have to be able to accurately pass the ball across the screen. When you start a passing game, you'll see the start point as a blue dot on screen. There is a rotating ball and five light coloured balls, which are the targets for your passes. Just make the player run to the rotating ball and he will pass it to the next point on screen. Run to it again and repeat the passing until the last ball has been reached, after which you can score a goal by dribbling the ball to the goal.

8) THE OPTIONS SCREEN

This screen consists of five option fields and three buttons. WE'll explain what each of them does.

8.1) Game/Practise Type

You can choose between 'Fetch' and 'Traject' (Oakey Dokie) or 'Dribbling' and 'Passing'. The differences are explained in chapters 6 and 7 of this manual.

8.2) Movement Control

This option lets you choose between two different ways in which your action figure (Oakey or the football player) responds to the joystick. Try both to find out which you prefer!

* Velocity Based: the action figure moves across the screen in direction the joystick points. His speed is dependant on the angle that the joystick makes, so if you just push the joystick a little, the figure will move slowly. If you hold the joystick in a certain angle, the figure on screen will still keep moving in the direction of the joystick.
* X/Y Based: the action figure moves to the same position of the joystick. If the joystick is centred, the figure will also stand still in the centre of the screen. If you move the joystick to one of its extreme corners, the figure will instantly go to the corresponding corner of the screen. This movement type is much faster that the velocity based one.

8.3) Force Feedback

* Click on the 'Enabled' checkbox (the white square or the text) to allow the joystick to assist you in the game. This option only works if you play the game with a force feedback joystick.
* The trackbar 'Level' allows you to adjust the amount of force that the joystick generates. The value can be set from -25 to 25, where negative values cause resistive force (that means the joystick pushes itself in the opposite direction you are supposed to push it) and positive values assistive force (the joystick helps you by pushing itself in the right direction). Beginning users with cerebral palsy will benefit most from a setting of 25 (maximum assistive force). More experienced users will find it challenging to disable the force feedback or set the level to a negative value.
* The 'Dead Zone' trackbar sets the area around the target where no force feedback is used. If set to zero, the force will be present at any place on the screen, in a linear relation with the distance from the monkey/player to the target. The maximum Dead Zone setting is 25, in which the user will feel no force in a fairly large area around the target. This setting may be adjusted to have the joystick exert force only when the target is far away.
* Click on the 'Time Interval' checkbox to enable a period after

which the force takes place.

* The trackbar 'Time Interval' sets the period after which the force takes place. It can be set from 0 (short) to 25 (long). Beginning users are advised to disable the time interval, so the force feedback is present all the time.

8.4) Difficulty

* Moving Target: click on this checkbox to make the banana or ball move across the screen, making it harder to capture but easier to spot.
* Highlight Target: this option enables a circle that will appear around the next banana/ball to be captured, or around the basket/goal if all targets have been caught. Checking this will make it easier to spot the target.
* Show traject: this option will make a yellow line to appear on the screen towards all targets. This makes it easier to determine the direction of movement for the monkey or football player. This option is especially recommended if the Game/Practice Type is set to 'Traject' or 'Passing'.
* Duration: you can set the duration of the game by changing the number of minutes. Standard this is set to 2. It is recommended to leave this setting as it is, for larger values will cause the user to tire soon. Also note that the highscore is not dependant on the game duration, so longer game duration will result in higher scores.
* Time Interval: click on this checkbox to enable a time interval in which you have to reach the next target. This will make the game more difficult to play. If the target has not been reached in the interval, no points will be added to your score and the game proceeds to the next target.
* The trackbar 'Time Interval' sets the period in which one trial has to be completed. It can be set from 25 (long, easy) to 0 (short, hard). Beginning users are advised to disable the time interval, so there is no pressure on reaching the target.

8.5) Previous Games

* Highscore: the highest score you have achieved in all your previous games. If you reach a higher score that this during play, a field with 'Highscore' will appear over your score.
* Games played: this number shows the total number of games

you have played.

8.6) Buttons

* ?: opens this manual.
* Start: clicking this button will start a new game with the settings you chose.
* Reset: all your settings are saved when you exit the game. If you click on the 'Reset' button, the game will load all your old settings from this file.
* Quit: exit the game.

9) CREDITS

Oakey Dokie 2.0 was created with Borland Delphi 5.
http://borland.com

10) CONTACT

Your feedback is very important for the further development of Oakey Dokie. For bug reports, comments, suggestions or complaints, please send me an e-mail.
Website of Bas Geerdink: http://Gdink.com
E-mail: Bas@Gdink.com

11) COPYRIGHT, TRADEMARK AND OTHER LEGAL NOTICES

References

[1] Becher, J.G.S.J.S.M. (1988). Klassificatie van stoornissen bij spasticiteit. Spastische en slappe verlamming van hand en voet; oorzaak, gevolg en behandeling.

[2] Berg-Emons van den, H.J.G. (1996). Physical training of school children with spastic cerebral palsy.

[3] Bobath, K. (1966). The Motor Deficit in Patients with Cerebral Palsy. Clinics in Developmental Medicine, No. 23.

[4] Eidinova, M.B., Pravdina-Vinarskaya, Ye.N. (1963). Cerebral Palsy in Children and its Treatment.

[5] Ingram, T.T.S. (1966). The neurology of cerebral palsy. Arch. Dis. Child., No. 41, pp.337-356.

[6] Keats, S. (1970). Operative Orthopedics in Cerebral Palsy.

[7] Lance, J.W. (1980). Symposium synopsis. R.G. Feldman, R.R. Young, & W.P. Koella (Eds.), Spasticity: disordered motor control, pp. 485-494, Chicago: Year Book Medical Publications.

[8] Lathan, C.E. (2001). A New Robotic Inter-face for Telerehabilitation. Proceedings of the State of the Science Conference on Telerehabilitation, pp. 60-63.

[9] Lathan, C.E., Malley, S. (2001). Develop-ment of a New Robotic Interface for Telerehabilitation. Proceedings of the 2001 EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing: Providing for the Elderly, pp.80-83.

[10] Levitt, S. (1962). Physiotherapy in Cerebral Palsy: A Handbook.

[11] Miller, Freeman, Bachrach, S.J. (1998). Cerebral Palsy; A Complete Guide for Caregiving.

[12] National Institute of Neurological Disorders and Stroke. (2002). Online: http://www.1uphealth.com/medical/disease/brainneurological-disease/cerebral-palsy-11.html.

[13] Opelousas Area Cerebral Palsy Clinic. (2003). Online: http://www.cpclinic.org/treatments.php.

[14] Reinkensmeyer, D.J., Painter, C., Yang, S., Abbey, E., Kaino, B. (2000). An Internet-Based, Force-Feedback Rehabilitation System for Arm Movement after Brain Injury. Report of CSUN's 15th Annual International Conference, "Technology and Persons with Disabilities", March 20-25, Los Angeles, CA.

[15] Reinkensmeyer, D.J. (2001). Java Therapy: A Web-Based System for Mass-Delivered Movement Rehabilitation After Stroke. Proceedings of the State of the Science Conference on Telerehabilitation, pp. 70-73.

[16] Reinkensmeyer, D.J., Pang, C., Nessler, J., Painter, C. (2002). Web-Based Tele-rehabilitation for the Upper Extremity After Stroke. IEEE Transactions on Neural Systems and Rehabilitation Engineering, Vol. 10, No. 2, pp. 102-108.

[17] Thiel van, E. (2001). (Dis)ordered motor control: Characterizing arm movements in hemiparetic cerebral palsy.

[18] Thorpe, D.E., Reilly, M., Case, L., Glenn, A.B., Hubbard, M.L., Ollendick, K., Petersen, K.E. (2001). The effects of aquatic resistive exercise on strength, balance, energy expenditure, functional mobility and perceived competence in persons with cerebral palsy.

[19] United Cerebral Palsy Information Source. (2002). Online: http://www.united-cerebral-palsy-information-source.com.