Silesian University Of Technology

Faculty Of Automatic Control, Electronics and Computer Science

Final Project

The use of a three-dimensional haptic manipulator-controlled gaming in the treatment of developmental coordination disorder

Author: Anna Wałach Supervisor: dr inż. Agnieszka Szczęsna

Gliwice, January 2014

Contents

Chapter 1

Introduction

In the modern world, physicians are able to successfully identify and treat significantly more illnesses and disorders than a century ago. New break-through ideas in genetics and neurology help to diagnose and prevent sicknesses. Development in imaging techniques facilitates great advance in research on functioning of human body and identification of disorders.

Cutting-edge technologies are crucial in the field of rehabilitation and surgery. New artificial limbs are being developed in order to better resemble functions of its biological original. Access to robotic surgical system enables physicians to perform far more precise and difficult surgeries and procedures than ever. Because of the usage of advanced graphical tools and 3D-printers it is possible to create more accurate prostheses for patients who suffered severe body damage.

1.1 Haptic devices

Specialized devices are being developed to resemble real-world reaction and be able to precisely identify operators' movements and intentions. The kind that focuses on generating real-time responses is called haptic devices. The most important feature of this type of controller is a tactile feedback produced by it that render the feeling of the texture and shape of objects shown on the screen of the computer.

Haptic devices are widely used in medicine. They serve as controllers for advanced graphical tools in which one can model precise and customised prostheses. Special applications are designed for beginner dentists and other physicians in order to provide opportunity to develop and practice their skills. For example, process of identifying cavity or performing a precise biopsy can be iterated as long as necessary without endangering the patient. This solution brings richer experience to professional training with lower cost and in safe, comfortable conditions. While the physician uses the stylus as a surgical instrument, the device provides realistic tactile feedback, which can resemble tissue softness or skeleton structure. The whole process is visualised as a graphical model by the application.

1.2 Dyspraxia

Studies show increasing number of neurodevelopmental disorders being diagnosed among children and adults. One of them is dyspraxia, also known as developmental coordination disorder (DCD), an inability to plan, organise and coordinate movement [7]. Depending on the level of severity, it can be a reason of small difficulties in everyday activities, for example fastening buttons or shoelaces, but may also make one unable to drive a car because of affecting working memory and coordination skills.

There are several various difficulties that occur during treatment of DCD. One of the most significant obstacles is a problem with conducting proper diagnose. Another - existing systems of therapy, not all of them equally effective, but all requiring experts in rehabilitation or psychology and hours of exercises with professionals. These problems will be addressed in chapter 2: Analysis.

DCD is very often observed among children in primary school. The most efficient way of treatment for this age group seems to be the "serious game". It allows to develop the necessary coordination or mental skills. Disguising the exercise as a game makes it more entertaining and pleasurable for a child.

1.3 Project goal

The main goal of this project is to create a coordination improving serious game, controlled by 3-dimensional haptic device that could be an effective tool in treatment of children with dyspraxia.

1.4 Report content

This report consists of the following chapters:

- Analysis - consists of:
  - a detailed description of dyspraxia and its treatment,
  - usage of a haptic device in rehabilitation,
  - significance of selected haptic factors in medical use of the software,
- Design - non-technical description of project requirements and game logic,
- Implementation - includes documentation, information about project composition, data structures and selected programming tools,
- Testing and launching - includes user's manual, launching examples and testing of application,
- Final conclusions - sums up the report.

Chapter 2

Analysis

Before describing the possible solution for the problem of rehabilitation for children with dyspraxia, an introduction about the disorder itself as well as its treatment methods and difficulties connected with them will be made.

2.1 Developmental coordination disorder (dyspraxia)

Developmental Coordination Disorder (DCD), also known as *developmental dyspraxia*, is the inability to plan, organise and coordinate movement [7]. It results in fine and gross motor problems and/or speech difficulties, which can have strong negative impact on daily activities and academic achievements. Dyspraxia was first described by WHO in 1992. It is a hard to diagnose and even harder to cure motor disorder, which is subcategory of neurodevelopmental disorder – along with autism and Down syndrome - and mostly identified among children. It often appears together with neurological illnesses from other subcategories, like ADHD, Asperger syndrome or dyslexia [6]. It is, however, also sporadically detected in completely healthy (except for DCD) children who are developing well intellectually [6]. DCD is not caused by motor or sensory impairments, similar to muscular dystrophy or Parkinson's disease. No correlation to any known neurological condition or intellectual impairment could be found.

2.2 Prevalence

DCD's prevalence range from 5% to 15% in the primary school and it is confirmed that about 5% of all children are affected, but not all of them have a severe case of DCD [6]. Children born prematurely [2] and children with extremely low birth weights [9] are at a significantly increased risk of having DCD. According to the majority of researches prevalence ratio is about 4:1 in boys than girls, but it is not confirmed - it differs between 5:1 to even 2:1 depending on study [6]. There are hypotheses that so high and unproven proportion is connected with the fact, that the frequent presence of comorbid conditions (e.g. ADHD) can make DCD easier to identify among boys. Also, girls do not have so strict social requirements regarding motor skills – therefore being clumsy may not be perceived as a disorder. Moreover, the majority of the children do not "grow out of" this disorder and it persists into adulthood.

2.3 Symptoms

There are numerous signs and symptoms of developmental dyspraxia as each case can be very distinct. The most noticeable ones are those connected with coordination and movement. Typically, representative symptoms of DCD are:

- difficulty in combining movements into controlled sequence,
- remembering the next movement in sequence,
- poor timing and disturbed balance.

The common cause for all of the above and other movement problems is working memory dysfunction. Working memory, often used synonymously with short-term memory, is the system in the brain which manipulates all "current data" - visual images or verbal information - and coordinates the central executive. One of the crucial responsibilities of the latter is creating a visual representation of possible movement paths. If this subsystem does not work properly, the patient will not perform his movements accurately.

Problems with working memory may have deep impact on patient's abilities and skills. It may result in having difficulties remembering deadlines and sequences of instructions, like cooking, driving a car or playing complex games. However, this disorder does not affect one's long-term memory, which actually can be excellent, despite poor short-time memory.

There are many more possible symptoms of DCD:

- problems with spatial awareness,
- determining left or right,
- estimating the distance to objects.

Fine motor problems can occur, i.e. difficulty in acquisition of graphemes (e.g. letters or numbers – that is why people with dyspraxia also sometimes suffer from dysgraphia or dyslexia) or establishing an efficient pencil grip.

2.4 Real-life impact

To conclude the description of disease, it is crucial to determine its impact on real-life activities. Issues with motor coordination may cause problems with walking, running, climbing or jumping. Working memory problems can lead to having difficulties driving a car, moving in crowded places, crossing roads and make a person prone to panic attacks. People with DCD are facing difficulties using cutlery, fastening buttons or shoelaces, shaving and performing similar routine activities. In conclusion, if the disorder is not cured, it may be extremely demanding on the patient.

2.5 Diagnosing

Although the disease is strictly neurological, researches about detecting DCD with imaging techniques like computed tomography, magnetic resonance imaging or ultrasound are inconclusive [5]. Several theories speculate that the etiology of DCD is part of the continuum of cerebral palsy; is secondary to prenatal, perinatal, or neonatal insult; or is secondary to neuronal damage at the cellular level in the neurotransmitter or receptor systems [8]. Experiments and researches in this topic are ambiguous and cannot be a basis for definitive diagnosis.

The most popular and conclusive way of diagnosing DCD is to perform a standardized motor test. A score below 15th percentile and IQ above 69 points qualifies the patient for a diagnosis of DCD [10]. However, there is some inconsistency among tests. In one study [11], two tests were compared - the Bruininks-Oseretsky Test of Motor Proficiency (BOTMP) and the Movement Assesment Battery for Children(M-ABC). The tested group consists of children with DCD and with no motor difficulties in 1:1 proportion. The size of test group was 312 in the first trial and 202 in the second. The test results shown respectively 82% and 67% consistency in distinguishing children who had DCD from children who did not [8]. Those are two of the most commonly used tests for diagnosing DCD - the potential lack of consistency in identifying DCD among children is a serious concern.

In the majority of cases, the diagnosis is based on observation of signs and symptoms. The American Psychiatric Association considers that DCD should be diagnosed only if the following four diagnostic components are present [3]:

- Motor coordination during daily activities should be substantially below that expected for age and intelligence.
- Resulting motor difficulties interfere with academic achievement or activities of daily living.
- The coordination problems are not due to a general medical condition (e.g., cerebral palsy or muscular dystrophy) or a pervasive developmental disorder.
- If mental retardation is present, the motor difficulties are in excess of those usually associated with mental retardation.

2.6 Treatment

The treatment for DCD is based on rehabilitation and training. No pharmacological therapy was designed to be used during DCD's treatment. Medications are reserved for the treatment of associated conditions (e.g., attention deficit hyperactivity disorder[ADHD]) [12].

There are two basic approaches of rehabilitation procedure - top-down and bottom-up. Top-down approach, also known as task-oriented or modular, attempts to remedy or improve specific difficulties by employing specific techniques aimed at the observed motor challenge (e.q., difficulty with handwriting, catching a ball, or performing fine motor tasks with the fingers). It usually involves gradually targeting certain problem behaviours and implementing step-by-step interventions that focus on teaching and practising the skill [12]. The most important element of modular-approach is practice. Practical applications of this method are therapies such as the cognitive motor approach with task orientation or task-oriented approaches with motor learning.

Bottom-up approach (process-oriented, deficit-oriented), the more global or generalised one is based on the theoretical assumption that the motor skills problem is just a manifestation of some underlying mechanism, such as impaired sensory integration or insufficient or inaccurate kinaesthetic perceptions. In the bottom-up approach, the therapist does not initially address the observable motor challenge. Rather, the expert focuses on how children manage their bodies, process stimulation (sensory information), and deal with problems. The expectation is that the improved sensory-motor (sensorimotor) functioning becomes generalized and eventually improves the motor skills. As children become comfortable with their bodies, they gain control of their motor (and other) functions [12]. This method is represented by the kinaesthetic training approach, sensorimotor integration therapy and sensory integration therapy.

No researches show that any of this approaches is definitely effective as DCD's treatment. This is probably caused by great diversity in symptoms and severity's levels of DCD. However, data suggest that top-down approach may be more efficient than bottom-up.

2.7 Haptic devices in rehabilitation

There are researches that prove the sensorimotor control of limb stiffness and compliance is a key element in the organisation of movement. Though, the best training approach would be the sensorimotor training. The goal of this therapy is to improve child's perceptual abilities intrinsic to the concurrent generation and experience of their own movements as well as use of this information to guide their movements [1].

This kind of training is difficult to apply with traditional therapeutic tools. However, various robot-assisted therapies are developed in order to support sensorimotor training, some of them including haptic devices. Robot-assisted therapies allowed to perform rehabilitation without the direct supervision of a trained therapist, what is especially important in DCD, where the number of patients significantly outgrows number of therapists. What's more, the complexity or/and difficulty of the task can be controlled very precisely.

Haptic devices were used to improve the generation of handwriting movements of children with DCD [13]. The researches show that this method can be notably effective, mostly because of active sensorimotor generation and control of movement trajectories. This is the feature required for learning generalized to task-related movements other than those specifically practised [1]. This requirement is caused by "catch-22" situation connected with movement learning mechanism. This learning process for a person without dyspraxia consists of two stages. First one is learning to generate imprecise approximation of movement.

When this is achieved, one can gradually improve approximation of a movement through practice until achieving optimal route. For people with dyspraxia it is almost impossible to generate generate good enough qualitative approximation, which should allow them to make quantitative improvements through practice - this is a "catch-22".

It is possible to overcome this paradox only if the support from the machine is balanced with possibility of active generation and control of movement from child. Mechanical properties of haptic devices, like inertia and tactile feedback can significantly help in achieving this result [1].

2.8 Haptic factors essential for rehabilitation. Phantom Omni description.

Figure 2.1: Sensable Phantom Omni device

Using haptic devices as controllers in application allows to benefit from the various haptic factors generated by particular haptic device. The factors vary between products and producers. Because all of quotable studies operate on Sensable Phantom Omni(Geomagic Touch Haptic Device) [14], when "haptic device" will be mentioned in text, it will always refer to Phantom device.

Phantom Omni (2.8 - Haptic factors essential for rehabilitation. Phantom Omni description) consists of stylus connected to an arm mounted on base. It has 6 degrees of freedom and its workspace range is 160x120x70mm. Thanks to its pen-like stylus it is easy and natural to operate it, especially for children. Phantom Omni has been widely used in industry for a several years, also for medical purposes, thus it is well tested and safe-in-use device.

The haptic factors generated by this device and tested in children's rehabilitation are [13] [1]:

- inertia,
- viscosity (of the space),
- magnetic attraction of the object),
- friction (of the object).

Viscosity and inertia may provide an appropriate support, but do not allow to keep track of designed movement form. Furthermore, no evidence suggest that the method could generalize the performance without support from the device [13]. Magnetic attraction help in following the generated path and this is the main haptic factor that will be used in the project. Researches show that friction has no significant impact on helping the child in performing the task [1].

2.9 Game rules analysis

In order to develop a serious game which can help children with dyspraxia, some basic rules and presumption must be established. The following principles are based either on [1], where similar project was developed or on the top of previous subsections analysis:

1. The game represents a bottom-up approach with sensorimotor training.
2. Basic task is to follow the path with tip of stylus, where the path is magnetically attractive.
3. The main haptic factor is magnetic attraction.
4. There must exists a levelling system.

The next chapter describes the design phase of a game based on this rules.

Chapter 3

Design

The design phase is focused on game logic construction. It is based on previous chapters analysis and discoveries from articles connected with the topic of haptic rehabilitation for DCD, especially [1].

3.1 Basic game rules and objects

As it was previously described in *2.9 - Game rules analysis*, the main task performed by patients is to follow the three-dimensional wire with tip of stylus, where the path is magnetically attractive. There is an object that player has to move from "start" to "end" along the path by pushing it with the tip of stylus, represented by cursor. There is also a competitor's object, moving along the path with uniform velocity (specified for each curve). The player wins a round when his object reaches the end before competitor's one.

3.1.1 Path

The mathematical representation of this path is Bezier's curve. Depending on level of difficulty there are different curves - they vary in length, curvature and torsion. The are three designed curves. Each curve has defined two times of passing - one for slower and one for faster race. The competitor's speed is calculated based on this time and the current path length.

3.1.2 Haptic objects and displayed objects

Terms "haptic" or "displayed" are used to describe object's visibility for haptic device. Only some of game's objects are rendered to haptic space and can be detected by the player while moving stylus through the space. Those are called haptic objects.

The haptic objects are:

- the path,
- the player's object.

None of the other objects (i.e. competitor's object, "start" sign) is available to be detected by touch, they are only displayed on the screen. All of the haptic objects are also displayed one.

3.2 Levelling up

There are three parameters that are changing along the game:

- the competitor's velocity,
- the path's shape (and length),
- the magnetic attraction level.

The player advances to the next level when he wins two times in a row with definite settings. While advancing, firstly, the competitor's velocity is increased. Then, when the player wins with both competitors, slow and fast one, he change the path to more advance, but with slower competitor again. Finally, after winning games on all paths, the magnetic attraction level is decreased and player starts from the very first path with slower competitor. This process is repeated until reaching the lowest possible level of magnetic attraction and is presented in *3.1.2 - Haptic objects and displayed objects*.

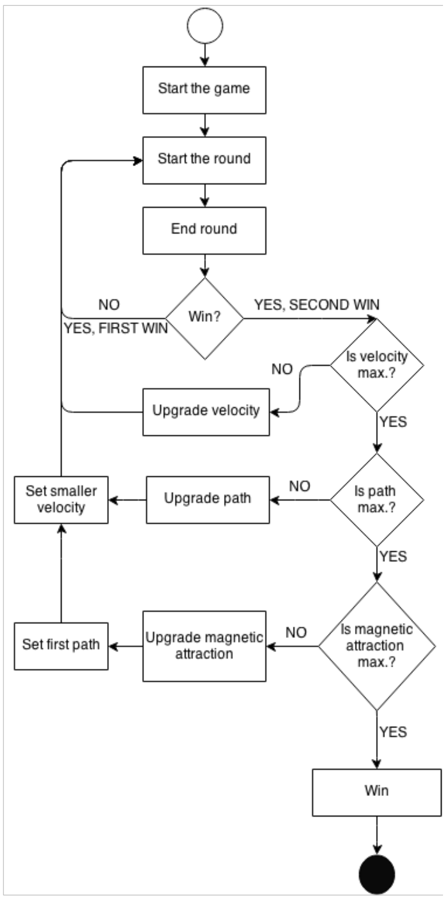

Figure 3.1: Schema of levelling process

3.3 Controls

There are three possible input devices: a keyboard, a mouse and the Phantom Omni. The player's object is controlled only with a Phantom Omni to avoid the possible confusion. The mouse allows changing of camera position. The keyboard is reserved for debug operations (turning on and off scene's objects, immediate winning of round).

3.4 Display and scene

The scene is closed in a skybox, a cube where inner walls are covered with textures. The path is rendered in the middle of the scene. Player's and competitor's object are attached to the path, so do start/end sign. There is also a graphic representation of a tip of a stylus.

3.5 Measuring player's score

Continuous measuring player's score during a game is one of the most important parts of the project, as it allows to control his progress. There are two main values measured during the game: *path length* (of the player's object) and *total race time*.

The path length may differ from curve length in case player leaves the "magnetic field" of a wire. This affect more often children with dyspraxia. This parameter decreases while trainings and therefore is index of progress. The path length is sampled with approximately 50-60 Hz frequency.

Total race time is a parameter responsible for determining of win. The counting starts from the moment when player's object moves for a first time in a round and lasts until reaching the end point.

Chapter 4

Implementation

The project is implemented using Object Oriented Programming paradigm, where objects represent elements displayed on the screen, haptic elements and controllers. The application has two threads - one for graphic and one for haptic calculations and operations. Graphic thread is working as a loop with about 60 Hz frequency, where haptic's frequency is around 1000Hz.

4.1 Haptic devices

At it was mentioned in *2.8 : Haptic factors essential for rehabilitation. Phantom Omni description,* the haptice device chosen for this application is Sensable Phantom Omni. It was chosen for this project because of common usage of this device in similar experiments and projects [1].

4.2 Programming language and libraries

In order to create a game, there is a need to choose the two main libraries - graphic and haptic one. Additional dependencies like audio library or model loader won't be present in this version of the application. The programming language should be compatible with both of libraries.

4.2.1 Graphic library

The graphic library that has been chosen is **OpenGL**. The main reason for such a decision is support from haptic dependencies for this library. Another arguments for this choice are:

- portability
- personal experience

Another possibilities were: DirectX library and OpenGL-based haptic libraries. The latter was rejected because of poor manual available and the former - because of worse compatibility with haptic device.

4.2.2 Haptic library

Two most popular libraries for usage with Phantom Omni are Open Haptics (from Phantom's manufacturer) and Open Source H3DAPI(from SenseGraphics).

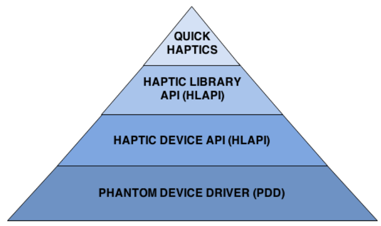

Figure 4.1: Description of Open Haptic APIs

Open Haptics is based on OpenGL using C/C++ and is available under two types of licence: academic and commercial. This library provides well written manual and programmer's guide as well as API description and examples. It consist of three separate APIs, each on different level of abstraction, as described in the figure *4.1 : Description of Open Haptic APIs*(the lowest level is the device driver). Quick Haptics is the most abstract API and allows quick development of application with limited user customisation. Haptic Device API is the lowest-level one and gives direct access to the haptic device together with possibility of independent generating haptic forces. Medium one, Haptic Library API, is used together with standard OpenGL applications as an "haptic addition", that allows to generate haptic objects from not-haptic user objects, but does not give such a low-level access as HDAPI.

Open Source H3DAPI is a software developed by SenseGraphics and its community. It is based on X3D, C++(OpenGL) and Python. It is high-level API that helps in haptic and graphic rendering. Unfortunately, the available manuals and documentation are very meagre. H3DAPI has its own application architecture, different than typical OpenGL application structure, so OpenGL code and experience cannot be reused while using this API. Application that uses Open Haptics can be developed separately by OpenGL developers and haptic developers. It is much harder with H3DAPI. Moreover, it is easier to teach OpenGL specialist Open Haptics than H3DAPI.

Then, the library of choice is **Haptic Library API** (under Academic Licence), because of following reasons:

- fast consolidation with working OpenGL applications
- well described documentation and examples
- level of abstraction low enough to perform minor customisation, but also high enough to avoid excessive control over haptic device

4.2.3 Programming language

Programming language used in the project is **C++**. It is prime language to use with OpenGL and is suggested language to use with Open Haptics. What is more, it allows necessary optimisation together with applying OOP paradigm.

4.2.4 Code samples

Some of the functions and solutions used in the application are based on single functions from Angel's OpenGL API and Sensable Open Haptics examples.

4.3 Other used programming tools

While developing an application, choosing the additional tools is as important as choosing libraries and APIs. The following tools are taken under consideration: IDE and version control system.

4.3.1 IDE

As the application is developed under Microsoft Windows 7 operating system, Microsoft Visual Studio 2010 is IDE of choice. It is advanced and convenient IDE for C++ programming under Windows OS. The version 2010 was chosen because of its stability and compatibility with Open Haptics as well as because of personal experience.

4.3.2 Version control

Distributed version system has been chosen, because of its clear advantage over not-distributed (each user has his own copy of full repository). The used version control system is Mercurial with TortoiseHg Windows client.

4.4 Components

The main application components dependencies are described in figure *4.2 : Components diagram*.

- *Controllers* - package name for *4.6.5 : Controllers*. Responsible for game logic and updating objects' properties.

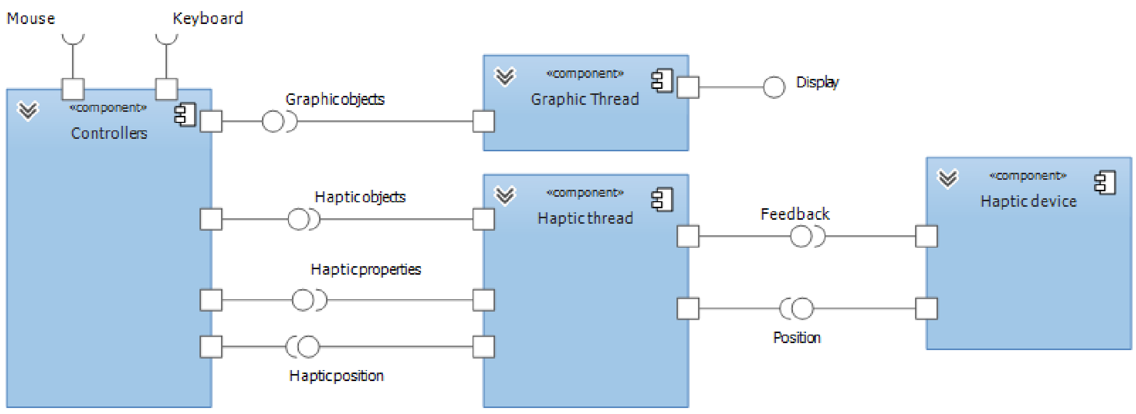

Figure 4.2: Components diagram

- *Haptic thread* - abstract name for both implemented and embedded functionalities of haptic API. Responsible for collision testing, generating feedback based on haptic objects' properties to device and passing the current position value.
- *Haptic device* - physical Phantom Omni device, receiving feedback parameters from application and sending back the current position.
- *Graphic thread* - abstract name for both implemented and embedded functionalities of graphic API. Responsible for rendering the display based on object's properties.
- *Mouse* - physical input from mouse, received by Controllers.
- *Keyboard* - physical input from keyboard, received by Controllers.

4.5 Data flow

Figure *4.3 : Data flow diagram* presents the flow of information in the application.

4.6 Class structure

Simplified class diagram is presented in figure *4.4 : Class diagram*. It presents the most important classes of the program and their dependencies. It does not contain all methods and fields - mathematical and some of private objects and functions are skipped in order to keep the diagram legible. Also, application uses some functions designed for graphical thread which are not presented in the diagram, but described in the following subsections.

4.6.1 Data structures

For application usage the following data structures were create:

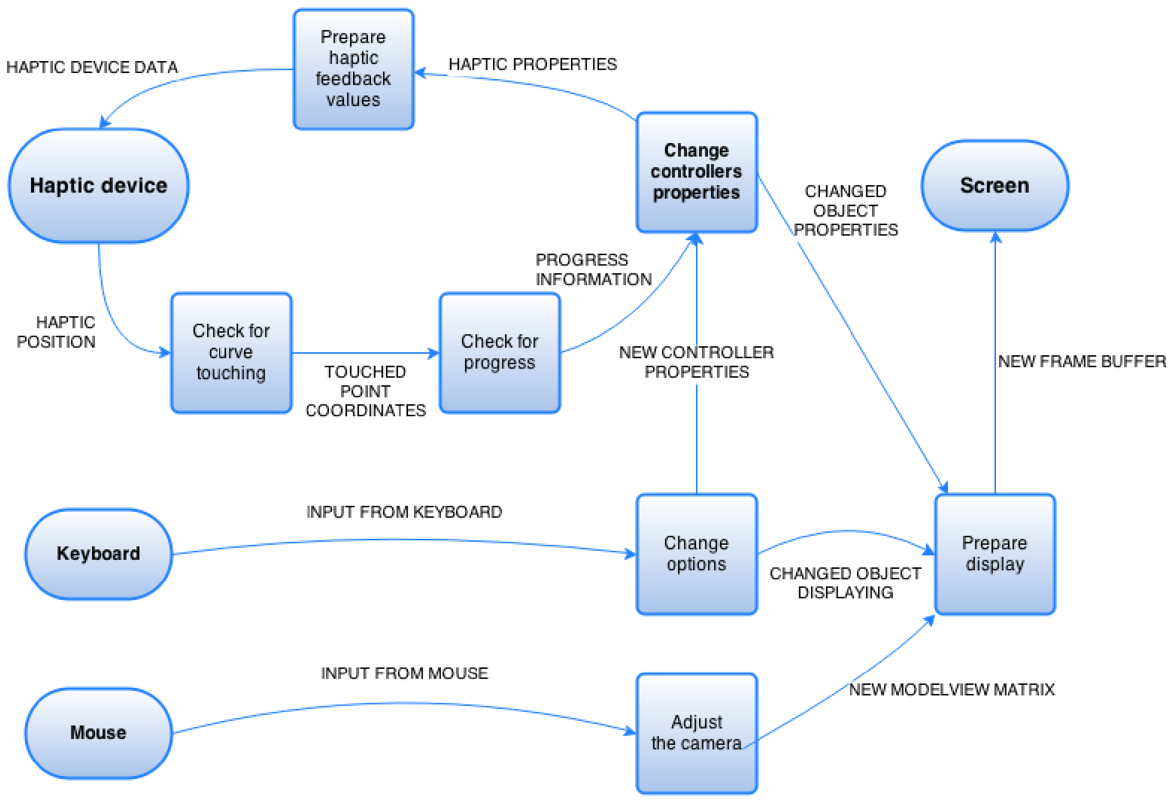

Figure 4.3: Data flow diagram

- *Curve* - storing path details like: control points (for Bezier's curve), total race time (slow and fast).
- *Skybox* - loads six skybox textures and keep skybox cube parameters.
- *Bait* - competitor's object.
- *Racer* - player's object
- *RouteController* - collects data about player's score.

4.6.2 Initialization

In order to properly initialize graphic and haptic thread, three functions were designed:

- *initGL()* - setting the lighting, defining OpenGL properties and initializing Skybox.
- *initHL()* - initializing the haptic device, setting proper callbacks for haptic objects.
- *initObject()* - initializing game objects.

4.6.3 Drawing components

The following components interact with graphic thread and draw to the graphic buffer:

- *CurveDrawer* - draws the path and/or path control points.

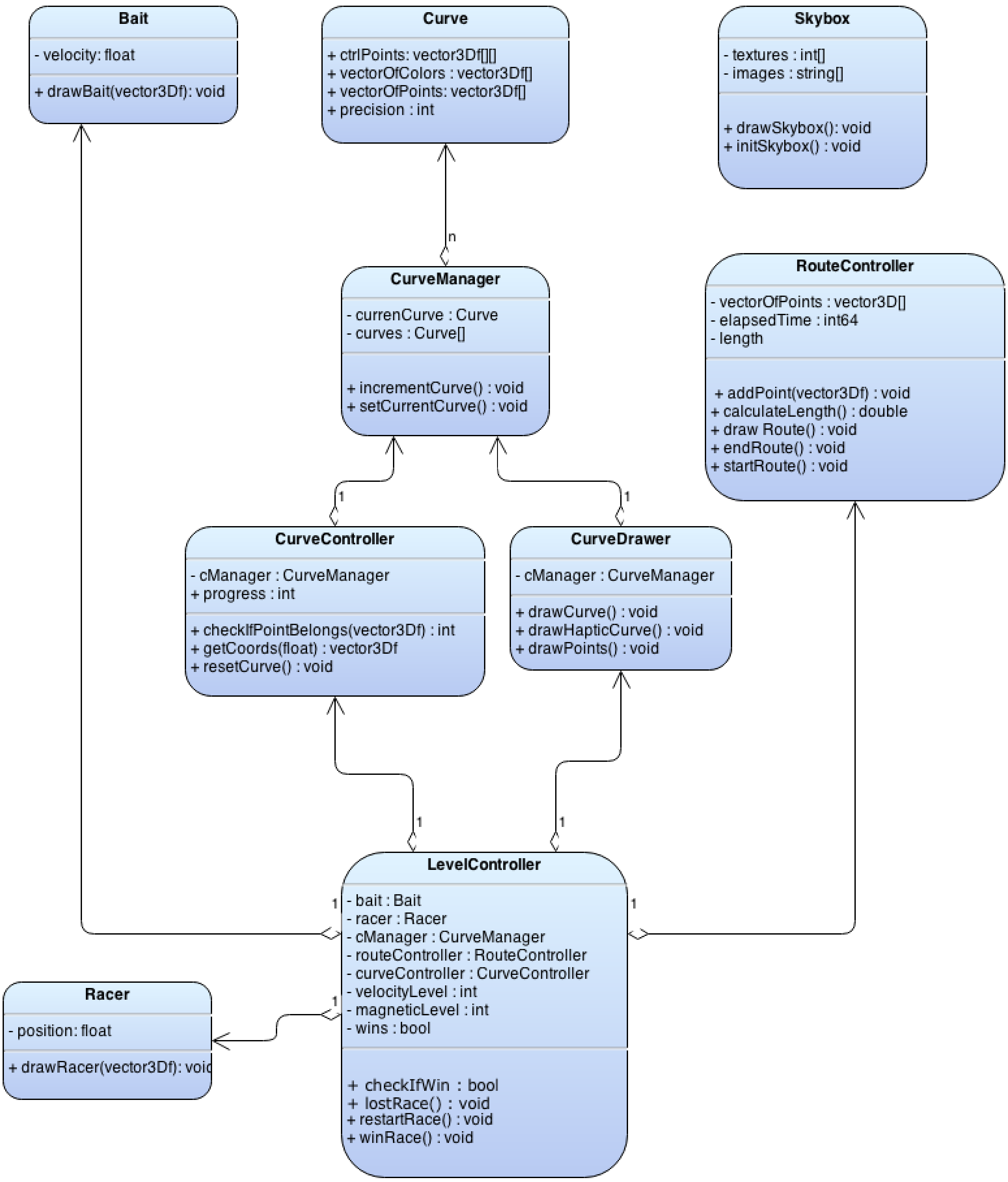

Figure 4.4: Class diagram

- *Bait* - draws the Bait.
- *Racer* - draws the Racer.
- *RouteController* - if proper mode is enabled, draws the player's route.
- *Skybox* - if proper mode is enabled, draws the Skybox.
- *drawCursor()* - draws cursor representation based on current proxy position.
- *drawSceneGraphics()* - calls all the other drawing functions and sets their properties.
- *drawMessage()* - draws message on the screen.

4.6.4 Haptic components

Haptic componenents are classes and functions that communicates with haptic thread and device.

- *CurveDrawer* - draws the path to the haptic buffer.
- *drawSceneHaptics()* - main function responsible for setting haptic properties and calling all the other functions that draws to haptic buffer.
- *updateHapricMapping()* - maps haptic device workspace to world coordinates after camera properties changes.
- *TouchCollisionThreadCallback()* - called when Curve is touched.
- *UnTouchCollisionThreadCallback()* - called when Curve stopped to be touched.

4.6.5 Controllers

Controllers represent the game logic.

- *CurveManagaer* - keeps track of current Curve and stores all the others.
- *CurveController* - checks for collisions with cursor and generate coordination based on 0-1 float number (0 - start of the Curve, 1 - end of the Curve).
- *RouteController* - measures the player's path and calculate length and time of race.
- *LevelController* - responsible for win checking, passing to another level and updating all the other controllers.
- *glutKeyboard()* - function responsible for grabbing input from keyboard and calling appropriate function when key pressed.
- *glutMouse()* - function responsible for detecting mouse buttons' pressing and setting appropriate control values.
- *glutMotion()* - function responsible for measuring mouse motion and setting appropriate camera values, if mouse buttons are pressed.

4.6.6 Camera

Camera is not represented as seperate class in program, however it has predefined global properties that can be changed with mouse. Camera properties are updated by keyboard and mouse functions and applied to modelView matrix in the following functions:

- *glutReshape()* - initialises modelView matrix and updates it after window reshaping.
- *updateCamera()* - updates modelView matrix after camera properties change.

4.7 Implementation details

This section is focused on parts of code that require additional explanation or description.

4.7.1 Path graphical and haptic representation

Path is depicted with Bezier's curve. However, while drawing it, application evaluates some finite number of points that belongs to curve and uses them to draw it. Number of points is defined by precision attribute of CurveController (its value is about 1000). Displayed path consists of small beads (spheres),where each bead represents one point of a curve. Haptic path representation is a line strip between those points. The reasons are simple:

1. Necessity for differences in precision value.
   The precision for haptic path is smaller than for displayed one in order to keep the path smooth in touch - haptic thread frequency is 1000Hz and if it has to test too high (from tests - around 500) number of objects, it acts like the surface it touches is rough and it is hard to move the stylus along the path. However, it its important to remember, that precision for haptic cannot be too low - the haptic and displayed path can differentiate in shape in such case.
2. Necessity for differences in points connecting method.
   The basic way of connecting the points is using OpenGL property GL_LINE_STRIP. While adding vertices (points) to a buffer of the path shape, each new point is connected with the last added by line. Obtained line has no depth - light has no influence on this shape, which means that it's color is uniform and give no clues about distance or depth. This problem can be solved if each point is represented as a shape - in this case, the sphere. Lighting works on spheres and moreover, this solution makes the path looks like a string of beads, which is much more attractive for children than casual line.

4.7.2 Progress detecting

Progress detecting is a process that repeat with 50 Hz frequency. It has to determine if the player is able to move his object or not by detecting the collision of path, cursor and the object.

The information, that player is touching the path, comes from haptic thread. It has appropriate properties set if such event is happening. The problem is in checking on which part of path play is now. The whole path cannot be checked during the short time (not even 20 ms) because it would require checking collision with even 500 elements that path consists of (the path construction was described in subsection above). The approximation is used, that in current moment of time application tests only 10 elements (points) that lied on the path just in front of the player's object - if the cursor "collide" with any of this points, object is moving to this point and procedure is repeated until end of path.

Chapter 5

User's manual

This manual describes game requirements and controls.

5.1 Launching prerequisites

- In order to run the application it is required to plug Phantom Omni Device (through 6-pin FireWire port) and install a proper, up-to-date driver.
- Suggested systems are Microsoft Windows XP/Vista/7 (32/64 bit).
- The graphic card should support OpenGL version (at least) 3.2.

5.2 General tips

General clues about playing the game are:

1. Round is started by touching the curve.
2. The precision of a player's movement is more important than it's speed. Moving too fast will cause not acquiring the points player passed(because of low sampling frequency).
3. Game loads new level(or reset the level) automatically.

5.3 Controls

Three main input devices that control the application are:

- *Phantom Omni* - controls the cursor during the game. There are two buttons on the stylus, but they have no influence on application.

- *mouse* - controls the camera. Moving while pressing left button allows to rotate the camera, right button - to translate, and middle one - to scale the view.
- *keyboard* - responsible for debug/advanced options, not available in normal game

The player should use only Phantom Omni to control the game and eventually the mouse to adjust the view. The rest should be controlled by the supervisor.

5.4 Advanced options

Advanced options are available only from keyboard. They allow to toggle some elements of the scene or use cheats. Possible short-cuts are:

- "s" - toggle skybox
- "r" - toggle the route view(of player's object)
- "p" - toggle the control points of Curve view
- "w" - immediate passage to next level
- "q" - quit the game

5.5 Quitting the game

Quitting the game is possible by closing the window or pressing the "q" key.

Chapter 6

Testing and launching

This chapter presents the snapshots from launching of the application and short description of testing methods and processes.

6.1 Examples of launching

The game can be launched in two modes - normal and debug. Normal presents enough functionality for children to play it, while debug mode offers additional functionalities for therapist and/or developer.

6.1.1 Normal mode

Figure 6.1: First curve (Lowest level).

In the figures *6.1, 6.2, 6.3* three paths from game are presented. White sphere is a start/end point. Blue sphere is the competitor's object. Yellow sphere is the player's object and the cursor is presented as a

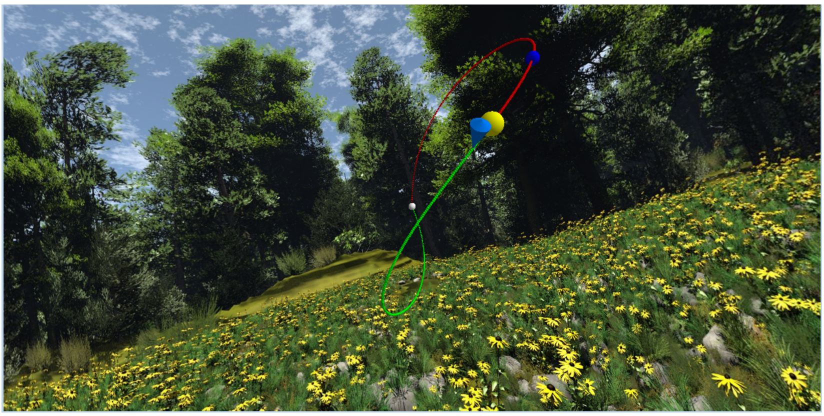

Figure 6.2: Second curve (Middle level).

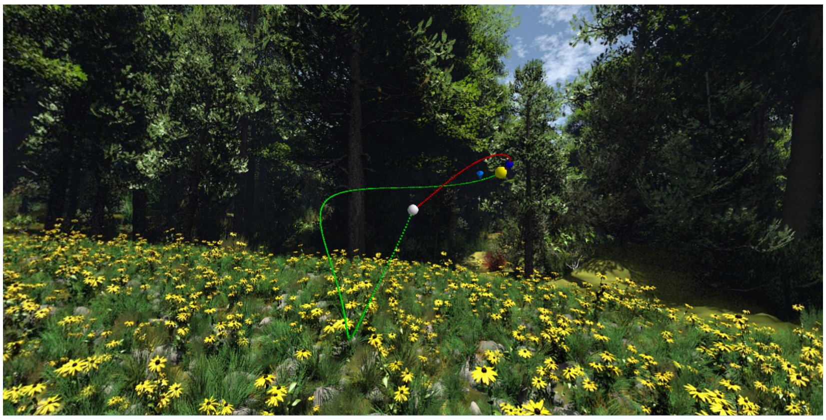

Figure 6.3: Third curve (Highest level).

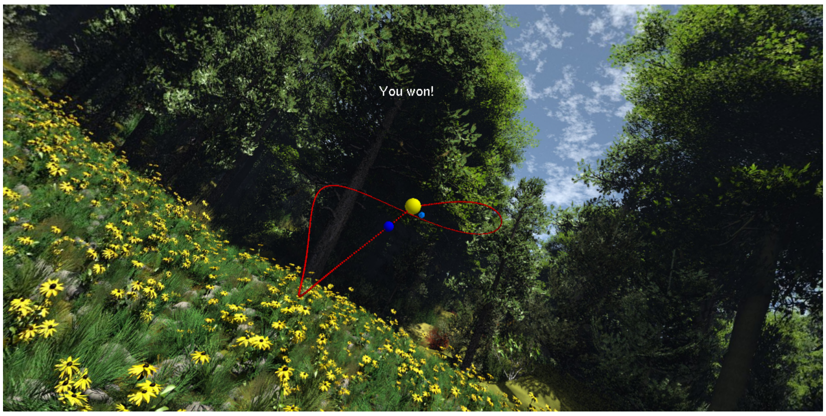

Figure 6.4: The game is displaying an information about the win.

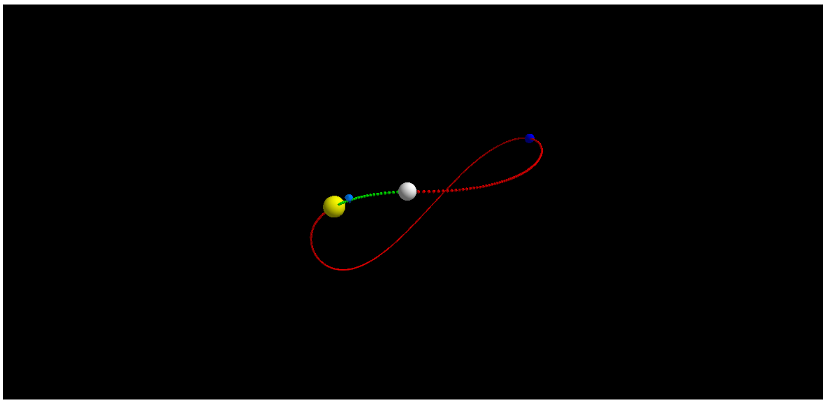

Figure 6.5: Scene view without the skybox.

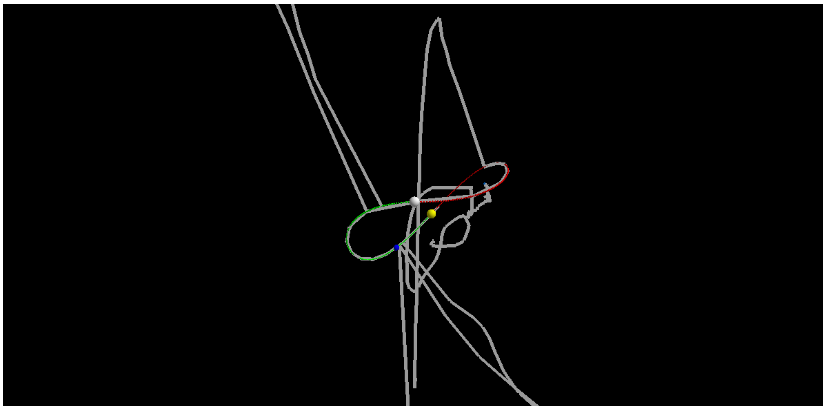

Figure 6.6: The route of the player's object.

blue cone. The passed distance is marked with green color. The message about winning and state of the game just after level restart is presented in the figure *6.4*

6.1.2 Debug mode

Figures *6.5*, *6.6*, *6.7* present different toggled debug option. In figure *6.6* the player's object real route is showed. The sharp impulses are effects of sudden getting out of the range of magnetic attraction field (it requires pressure which is hard to control when the field disappears).

6.2 Testing

Because developed application is a game, testing was focused on specification-based testing(requirements based on analysis and design phase) and visual testing. There was a small number of unit tests on first stage of development in order to check mathematical function correctness. The following subsections present screen-shots and descriptions of specification-based tests. Presented requirements are based on *3 : Design*.

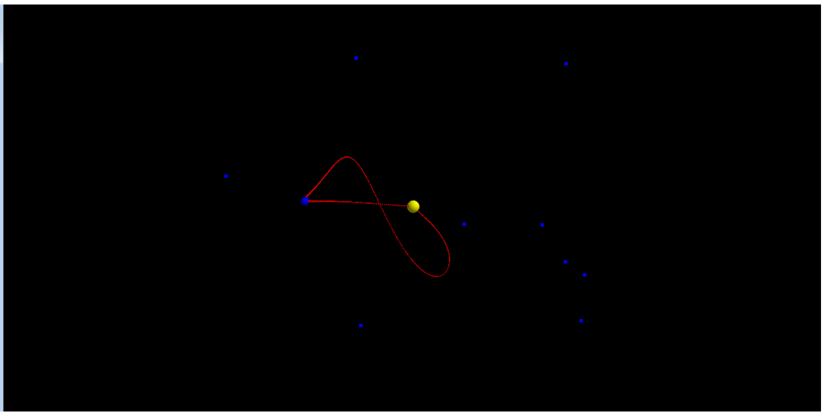

Figure 6.7: Control points of presented Bezier's curve.

6.2.1 Haptic thread

Testing of haptic thread is presented on the film attached on CD's. The following statements were tested positively:

- "Cursor should get attracted by path and be able to push the Bait."
- "Cursor should be able to get off the path."
- "The path should be smooth in touch and consistent with graphic representation."
  It is very important requirement, because display path and haptic path are two different shapes, as described in *4.7.1 : Path graphical and haptic representation*.

6.2.2 Game logic

Majority of the game logic is connected with levelling mechanism. The rest regards the problems of display and resetting and setting new objects properties before new round.

- "The player should level up when he moves the project from start to end in time equal or less than specified for this curve and competitor winning time."
  This requirement's testing can albo be seen on the film.
- "The message should show up when player advance to the next level."
  This requirement was tested positively in *6.4 : The game is displaying an information about the win*.
- "The Bait and Racer should start new round from start point"
  This requirement was tested positively in *6.4 : The game is displaying an information about the win*.

Chapter 7

Final conclusions

7.1 Project evaluation

All of the planned functionalities have been implemented. Application:

- displays a level (path, objects, cursor) properly.
- can correctly detect the progress
- has fully functioning levelling system.
- can correctly measure player's score and data

The game is ready to be presented to a children with dyspraxia as a rehabilitation tool. Although adding some more children-friendly features (like sounds, more colourful and friendly objects) may be considered to allows better interaction with patients, it is not required for application to work properly.

7.2 Similar project results

Similar application was developed by authors of [1]. Firstly, children's (with and without DCD) score in performing the task(similar to the one in game, but without magnetic attraction) was checked. Then, children were trained once a week for about 20 minutes with usage of application, up to five times. After this, they were tested again. Before the training, results of children with DCD were significantly worse than results of typically developing children. After the training, both groups improve, but the progress of children with dyspraxia was notably greater. It means that it is possible to overcome the "catch-22" syndrome and children with DCD can actually learn complex motor skills with proper support and training method.

7.3 Summary

The project was successful. All requirements have been implemented. Fully functional, stand-alone application has been created. The Open Haptics API was a good choice - developing application with this library was convenient and fast. It surely has great potential and can be use to create innovative applications that uses the Phantom Omni manipulator.

The next stage of the project should be testing it with children, so the game fulfils their requirements and than perform tests, where application would be used as a rehabilitation tool for children with dyspraxia.

Development Coordination Disorder is troublesome disease, also because of psychological and social reasons. Engaging in the development of treatmeant for this disorder was really valuable experience.

Bibliography

[1] Winona Snapp-Childs, Mark Mon-Williams, Geoffrey P. Bingham, *A Sensorimotor Approach to the Training of Manual Actions in Children With Developmental Coordination Disorder*, J Child Neurol. 2012

[2] Bowens A., Smith I. *Childhood dyspraxia: some issues for the NHS. Nuffield Portfolio Programme Report No: 2* Leeds, Nuffield Institute for Health 1999.

[3] *American Psychiatric Association Diagnostic and statistical manual of mental disorders.* 4th edn, text revision. Washington, DC: American Psychiatric Association, 2000.

[4] Polatajko H.J., Cantin N. *Developmental coordination disorder (Dyspraxia): an overview of the state of the art.* Semin Pediatr Neurol. 2006;12:250–258.

[5] Missiuna C., Gaines R., Soucie H., McLean J. *Parental questions about developmental coordination disorder: A synopsis of current evidence.* Paediatr Child Health (October 2006) 11 (8): 507–12

[6] Gordon N., McKinlay I. *Helping clumsy children.* Edinburgh: Churchill Livingstone, 1980.

[7] Polatajko H.J., Fox A. M., Missiuna C. *An international consensus on children with developmental coordination disorder.* Can J Occup Ther 1995. 623–6.6.

[8] Barnhart R.C., Davenport M.J., Epps S.B., Nordquist VM. *Developmental coordination disorder.* Phys Therapy (Aug 2003) 83 (8): 722–31.

[9] Holsti L., Grunau R.V., Whitfield M.F. *Developmental coordination disorder in extremely low birth weight children atnine years.* J Dev Behav Pediatr. 2002 Feb; 23(1):9-15.

[10] Geuze R.H., Jongmans M.J., Schoemaker M.M., Smits-Engelsman B.C. *Clinical and research diagnostic criteria for developmental coordination disorder: a review and discussion.* Hum Mov Sci.2001 ;20(1–2):7–47.

[11] Dewey D., Wilson B.N. *Developmental coordination disorder: what is it? Physical and Occupational Therapy in Pediatrics.* 2001 ;20:5–27.

[12] Stephen L. Nelson Jr., *Developmental Coordination Disorder.* 2013

[13] Ben-Pazi H., Ishihara A., Kukke S., Sanger T.D. *Increasing viscosity and inertia using a robotically controlled pen improves handwriting in children.* J Child Neurol. 2010;25:674-680.

[14] *Sensable Phantom Omni Product Page*, `http://geomagic.com/en/products/phantom-omni/overview`, January 2014

Appendix A

Added CD's desctiption

Załączona płyta CD zawiera:

- `main.pdf` - electronic version of this report
- `abstract.pdf` - the abstract of this report
- `/code/*` - source code of the application
- `/lib/*` - libraries used in the application
  - *glew 1.10.0* - lib and headers files
  - *freeglut 2.8.1* - lib and headers files