Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias da Informação

KSGphysio – Kinect Serious Game for Physiotherapy

Nuno Miguel Santos Duarte

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Engenharia de Telecomunicações e Informática

Orientador(a):

Dr. Octavian Postolache, Professor Auxiliar
ISCTE-IUL

Co-orientador(a):

Dr. Jacob Scharcanski, Professor Associado
Universidade Federal do Rio Grande do Sul

Novembro, 2014

Abstract

Serious games fall under a set of applications capable of improving recovery times by increasing the player's engagement. In this thesis we focus on the possibility of joining that capacity to Microsoft Kinect sensors ability to collect data without the need of additional sensors and present an application capable of giving proper feedback about the player's behaviour during a rehabilitation session. Using an Android OS mobile platform as interface acquired data, the proposed solution is a prototype that aims to facilitate the analysis of the patient’s progress during rehabilitation sessions based on serious games usage. Results associated with arm rehabilitation through serious game, as so as the mobile graphical user interface are included in the present work.

Resumo

Os jogos sérios inserem-se num conjunto de aplicações capazes de melhorar o tempo de recuperação, aumentando o envolvimento do jogador. Nesta dissertação centrámo-nos na possibilidade de unir essas capacidades com a capacidade do sensor Microsoft Kinect para recolher dados sem a necessidade de sensores adicionais e apresentar uma aplicação capaz de dar feedback adequado sobre o comportamento do jogador durante uma sessão de reabilitação. Usando uma plataforma móvel Android OS como interface entre os dados e o utilizador, a solução proposta é um protótipo que visa facilitar a análise da evolução do paciente durante as sessões de reabilitação com base na utilização de jogos sérios. Os resultados associados à reabilitação dos membros superiores através de um jogo sério, bem como as estatísticas apresentadas usando a aplicação móvel, estão incluídas no presente trabalho.

Acknowledgments

This work was supported by the Instituto de Telecomunicações and Fundação para a Ciência e Tecnologia and the project “EHR-Physio: Electronic Health Records-Needs, Requirements and Barriers in Physiotherapy”, PTDC/DTP-DFS/1661/2012 a special acknowledgement is granted.

Table of Contents

List of figures

List of Acronyms

VR	Virtual Reality
EHR	Electronic Health Record
API	Application Programmer Interface
IR	Infrared
FPS	Frames Per Second
DBMS	Database Management System
RDBMS	Relational Database Management System
PHP	Hypertext Processor
PDO	PHP Data Objects
SQL	Structured Query Language
JSON	Javascript Object Notation
UI	User Interface
HTTP	Hypertext Transfer Protocol
GUI	Graphical User Interface

1 Introduction

The use of a computer-simulated environment to simulate physical presence in places in the real world or imaginary worlds is a constant practice today. Virtual reality (VR) can create sensory experiences, including virtual taste, sight, smell, hearing and touch which can be used to create real-life scenarios that helps to improve therapy efficiency by creating greater engagement. The VR technology used in the majority of rehabilitation clinics is based on adapted games, which are regular video games that were originally developed for recreational use but by virtue of being able to allow the player to perform a plurality of tasks that resemble real situations, are used for rehabilitation purposes. However, because they are adapted games and they were not specifically designed for rehabilitation, many therapists prefer to just use them as a complement to a physical therapy session rather than as part of the rehabilitation session. Adapted video games are very limited because they are not configurable or accessories are needed to interact with it, which sometimes makes it impossible to use in a rehabilitation session of a more impaired user, do not allow tracking progress and in most cases there is a lack of feedback to the therapist and to the patients under rehabilitation process [1].

Research suggests that repeated execution of an exercise is sufficient to stimulate the brain to remodel and promote better motor control of the limbs [2]. However, the number of exercises performed during a rehabilitation session is not sufficient for this to happen, which suggests the need for patients to practice these exercises at home. However, 65% of patients self-reporting being non-adherent or partially adherent to their home programs, which in most cases represents a regression for the patient. Adherence embraces 2 elements: being compliant with the frequency of the suggested exercises and carrying out each exercise with the correct biomechanical alignment [3].

Another flaw in the physiotherapy domain is the lack of an history with the patient's previous motor or cognitive problems and collected data from previous rehabilitation sessions as occurs in other areas as general medicine, where this type of stored data is referred as Electronic Health Record (EHR) and describes a collection of a patient’s health and healthcare data. It is a record in digital format that is shared across different health care settings and also be shared by a network-connected system. Basically it is a cross-institutional information system containing all a subject's health information since he/she started using the medical resources available in society. It includes a set of data as medical history, medication and allergies, immunization status, laboratory test results, radiology images and vital signs that are relevant to a subject’s medical treatment but also to a subject’s health in general. To an EHR system the patient is regarded as an active partner in his/her treatment by accessing, adding, and managing health-related data, thereby supporting care [4].

1.1 Objectives and Contributions

This work aims to produce a system divided in three components to improve rehabilitation sessions quality in the clinic or at home, while enabling data collection and communication to track the patient physical rehabilitation progress and to extend the current Electronic Health Record system to physiotherapy.

The first component is a 3D game based on real life situation that combines the virtual reality ability to increase the player's engagement to the Microsoft Kinect sensor ability to collect data, without the need of additional accessories. This game has the distinct feature of being configurable, which means that it can be adapted to the patient's needs. The second component is a mobile application developed to Android OS devices, which behaves as an interface between the therapist and the data stored on the server. The mobile application deals with the collected data and shows it to the therapist through a set of charts and statistics that he/she can understand. The last component is an Application Programmer Interface (API) that eases the communication between the other components of the system and the database by dealing with the requests from the other two components.

A paper was written and successfully submitted to the 8ª Edition of the IEEE conference EPE 2014 – International Conference and Exposition on Electrical and Power Engineering. The paper can be found in Appendix E.

1.2 Thesis's Organization

Chapter 2 contains an overview of the work done in this area of investigation and how the Microsoft Kinect sensor can be used in physical rehabilitation. In chapter 3, the usefulness and benefits of serious games combined with Kinect sensor technology in physical rehabilitation is discussed. The description of the database implementation follows in chapter 4. The need, the implementation and the description of the created API are discussed in chapter 5. In chapter 6, the arguments that support the value of having a mobile application as interface between the collected data and the therapist are presented as well as the description of the mobile application. Finally the results of the experiments, conclusions and future work are presented in chapter 7 and chapter 8 respectively.

2 Related Works

Virtual reality is increasingly used in areas such as rehabilitation through motion-based games. This new generation of tools for rehabilitation has grown up faster in the past few years. However, many of these games require wearing a number of sensors attached to the body or use extra material to detect the movements of the patient [5,6].

2.1 Serious Games

Literature has shown that video games cannot only be used for fun and entertainment. They can be used to help people to improve their fitness and coordinative abilities by increasing patient engagement by making rehabilitation sessions more fun.

In [7] the impact of serious games on prevention and rehabilitation is discussed. Authors applied three criteria to describe the additional benefits of serious games: Effectivity and efficiency are the first; Social, psychological, physiological and sensory-motor factors are the second; Quality of study is the third. Using these criteria they conclude that serious games can produce substantial benefits at least for a considerable portion of patients. However, they suggest that game concepts and game interventions have to be developed and evaluated to adjust to the patient prerequisites because different people have different preferences and needs. Another key issue to be solved is sustainability. Serious games have been proved to produce transient effects, so to achieve sustainability, authors suggest that researchers have to evaluate which settings support long-term motivation and engagement for different people with different social, physiological, psychological and sensory-motor factors.

According to [8], where authors evaluate a set of commercial games with a group of therapists trying to identify the best game design principals with particular relevance to rehabilitation, the meaningful play and challenge are the best game designs. Meaningful play is the game design where the player's actions affect the system's outcome. Authors suggest that the recognition of the effect a particular action has in the game can increase the patient engagement. The challenge game design is a type of game where the user is challenged during the game time, more the player plays the game more his/her skills increase so the player requires a high level of challenge to continue enjoying playing.

In [9] the development and testing of a serious game based on movement therapy to encouraging stroke patients with upper limb motor disorders to practice physical exercises is described intending to show how serious games can contribute to motor therapy by providing more motivational tasks. Authors concluded that the integration of VR simulation with serious games adds richness to the virtual environment. Initial results shown that serious games intervention had a positive

impact on the recovery of patient's upper limbs movement with patients reporting that they enjoy playing the game.

2.2 Technologies based on Sensor Networks

For a long period of time, sensor networks have been identified as the most efficient solution in the area of rehabilitation session's motorization. Sensor networks can be divided into two categories: The wearable, which can be attached to the patient's body and the fixed or environment, which must be precisely placed in a room to track patient's movements and collect data. Usually, with this type of sensors the patient must have some markers attached to his body so that the environment cameras can detect his presence.

Environment sensor networks have proven over time to be a very efficient solution, though quite limited. In fact, these technologies are very expensive, difficult to move and a high level of expertise is required to work with it so, these technologies are usually stored in specialized laboratories and must be operated by qualified personnel. For those reasons environment sensor networks can't be a handy solution for monitoring rehabilitation sessions.

According to [10], over the years the advances made in the field of microelectronics have enabled researchers to develop miniature circuits that allow the aggregation of a large number of essential sensors for monitoring rehabilitation sessions such as gyroscopes, accelerometers, and pedometers in small and compact components making thus feasible the use of wearable sensor networks. In the field of wireless communication new standards that improve the interoperability and data transmission between sensors were created.

As explained in [11], wireless sensors networks have been used for a long time in clinics for monitoring rehabilitation sessions however, there are some disadvantages. The authors identify some key issues for the proper functioning of this type of systems, such as the fact that the members are composed of several segments connected together making necessary the use of several sensors to monitor each one of the patient's limbs during a rehabilitation session and it also makes all system integration and data crossing become more complex. Another problem identified by the authors was the impact of the human body on the propagation of signals, how the amplitudes overestimate the skin and acts as a low pass filter for the transmitted signals. It is even said that, for a correct use of such technologies, it is required the constant presence of a physiotherapist, which in the majority of practices is impossible because, usually, each therapist has more than one patient at a time.

2.3 Wii

Monitoring technologies for rehabilitation sessions based on Nintendo Wii have been very successful in terms of bringing an atmosphere of relaxation and encouragement around the users [1]. That represents a potentially effective and safe alternative to traditional rehabilitation techniques encouraging users to perform rehabilitation movements while playing virtual games [12].

As mentioned in [1], Nintendo Wii has been used by many physical therapists as a working tool because it is easier to use and have a wide variety of games available. Although there is a wide range of games available, authors say that not every game can be used for the purpose of rehabilitation because they were not designed or developed considering this particular function, creating some limitations to the use of this technology as a solution for monitoring rehabilitation sessions. Among that set of limitations the authors highlight the lack of precision in monitoring and tracking the evolution of the patient, the difficulties experienced by patients when playing certain games because they have been developed for healthy users and do not take in consideration possible physical or psychological limitations of the patient. Another problem described is the existence of very generic metrics, which are insufficient to carry out the monitoring of the evolution of the patient and get proper feedback.

Given these problems, authors propose an approach that combines a system based on Nintendo Wii with a web-based application that allow motion capture to monitoring exercises and track patient's progress.

In [12] authors test the effectiveness of Wii gaming technology in stroke rehabilitation combined with a 2-point infrared light sensor, mounted on top of a television, that compute and reproduces on the screen the movement from the controller as performed by patients. They use a set of commercial Wii games that that attempt to replicate daily tasks as playing sports or cooking meals. Authors suggest that the feedback provided by the television generates positive reinforcement, facilitates training and task improvement.

However, despite solutions presented above seem to give a proper feedback about patient's behaviour and help to improve rehabilitation sessions effectiveness, they were unable to solve the main problem in these approaches, which is the need of controllers and/or accessories to interact with the games. These controllers contain accelerometers that are sensitive to changes of direction, speed and acceleration and allow the patient to interact with the games [12]. This is an understandable limitation of these approaches because not all users are able to use controllers, and the need of a large variety of accessories to perform specific exercises also does not testify in favour of this approach.

2.4 Kinect

Applications based on Kinect sensors natural interaction without the need of additional sensors allow therapists to customize and adjust the physiotherapy training to be performed by the patient and give more freedom of movement.

According to [2], after testing with Kinect-based applications in different situations and in patients with different levels of rehabilitation, the authors concluded that the accuracy rate in the detection of motion using a Microsoft Kinect sensor is more than 80%. The reason they couldn't achieve the 100% accuracy was that the wheelchairs and walkers used by patients influenced the judgment of the Kinect. Nevertheless, therapists have rated the technology positively indicating that this kind of tool

would reduce their labouring burden and improve rehabilitation efficiency, while patients said that this technology has helped to increase their motivation to participate in rehabilitation reducing the time affected by this process.

In [13], authors show their concern about the lack of arguments that validate rigorously the technical performance of Kinect sensor as a rehabilitation tool, although previous works prove the potential of Kinect sensor in this area. They studied the trajectories of the joints at the right hand, right elbow, and right shoulder when performing motor task External Rotation, Scapular Retraction and Shoulder Abduction and compared the results of a Kinect sensor with the results of a OptiTrack optical system that is a marker-based system which requires users to wear reflective markers such that their movements can be tracked by an array of cameras. This system is known for its high precision and processing capability. They compared the outputs of the two systems after patients perform six different types of motor tasks that were previously selected and conclude that Kinect sensor can achieve competitive motion tracking performance as OptiTrack so it can be seen as a promising rehabilitation tool for use in the clinic or at home. They also evaluate the timing performance of both systems.

Through the analysis made in [14] of Kinect applications used in physical rehabilitation tests, using standard timed tests and joints tests. The first test was the timed "Up & Go Test" that is a reliable and valid test for quantifying functional mobility that can be used to follow clinical changes over time, the second timed test was the "10-Meter Walk" that is a test that evaluates the person's ability to walk using a start line and a end line with a Kinect sensors on each one and measures the time that the patient require to go from one point to another. For the joints test they used the "Range of Motion Measurement" that is a test used to measure the loss of mobility of the joints by measuring the angles of different body parts while performing different movements as abduction or extension considering the patient's posture and the examiner position. Authors present the results for these tests and foresee that Kinect sensor technology will be widely applied in medical care fields.

In [15], a low cost system that uses the potential of virtual reality combined with the capacities of the Kinect sensor and natural user interfaces to offer to patients with multiple sclerosis an intuitive and motivating way to perform motor rehabilitation exercises is tested. The system is presented as a tool that allows therapists to control the rehabilitation process and evaluate the progress of the patients. Tests are being made to find out the clinical utility of the system and the acceptance degree of patients.

2.5 Android Mobile Applications

Since the massification of mobile devices an IT application can have a mobile extension whether to access data or collect data. This approach allows data to be portable an accessed everywhere. The Android OS is one of the most used operating systems on mobile devices by virtue of being open-source. Most of the approaches in the physiotherapy field using mobile devices are centred in using the device as a tool to collect data.

As stated in [16], smartphones are seen as low-cost sensors that are easily accessible by majority of people, allowing the execution and monitoring of rehabilitation exercises anywhere. In this article, authors used a smartphone attached to a patient's limb and the signals generated by the accelerometer of the smartphone were stored by an application that was previously installed on the device. The application is configurable allowing each exercise to be customized considering the patient's need. They claim that by being configurable and enabling the patient to have a feedback of the work done during the rehabilitation session, this solution is significantly more enjoyable for users than traditional methods.

In [17], authors used the hardware capabilities of android-based devices to develop an application that turn the device in a wrist rehabilitation tool. They say that because this approach is running over an operating system that is capable of connecting to the Internet: the therapist can set up exercises considering the patient's needs and then the patient can upload them to his device, the movements can be reported back to the therapist to evaluate the effectiveness of the treatment and the system also has the ability to self-adapt to the patient, as the patient improves his performance of the exercises, providing a greater improvement in a shorter period of time.

3 Serious Game for Physiotherapy

The term "serious games" relates to digital games serving serious purposes like training, health, education or research. Beyond the traditional use of entertaining people, digital games have gained a new purpose when a new generation of games involving the use of sensors capable of detect whole-body movements (e.g. Kinect sensor) appeared. From that moment, this new generation of games played an important role in rehabilitation, enhancing health-related physical activity, improving sensory–motor coordination, preventing asthma, changing nutrition behaviour, alleviating diabetes and preventing smoking or HIV [7].

3.1 Kinect Sensor

Kinect is a motion-sensing device that was originally developed for Microsoft's Xbox360 gaming console. The main feature that distinguishes it among others in this gender is that it is not a hand-controlled device, but rather detects your body position, motion and voice. It replaces the controller that was once the heart of a gaming device by your body.

The idea of developing the Kinect sensor began to take shape when Microsoft engineers realized that the traditional game controller was the main barrier to making video gaming into a mainstream activity. They quickly realized that the solution was getting a device to track users' bodies as they move.

The core of Microsoft Kinect sensor technology came from a Israeli startup named PrimeSense that figured out how to encode patterns in the light beams and then measuring the changes in those patterns it's possible to give a particularly accurate view of a room [18]. The Kinect sensor has now outgrown its Xbox roots and is no longer limited to only gaming after Microsoft release Kinect for Windows, a version designed specifically for PC that helps developers to write their own code and develop real-life applications like serious games for rehabilitation purposes with human gestures and body motions.

The Kinect sensor includes a color camera, an infrared (IR) emitter, an IR depth sensor, a set of microphones and a LED, as seen in Figure 1. Additionally has a small motor working as the base that enables the device to be tilted in a horizontal direction. The color camera is responsible for capturing and streaming the color video data. The Kinect color stream supports a speed of 30 frames per second (FPS) at a resolution of 640x480 pixels, and a maximum resolution of 1280x960 pixels at up to 12 FPS. The IR emitter and the IR depth sensor work together to make things happen. The IR emitter constantly emits a infrared light in a "pseudo-random dot" pattern over everything in front of it, as seen in

Figure 2, and the depth sensor reads the reflection of the dotted light in the objects and converts them into depth information by measuring the distance between the sensor and the object from the IR dot was read from [19].

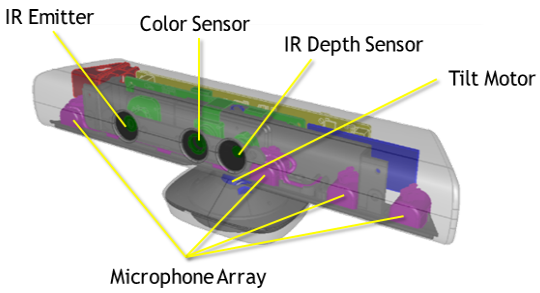


Figure 1- Microsoft Kinect sensor architecture

Figure 2 - Pseudo-random dot pattern

3.2 Game Engine

A game engine is an imperative tool when you want to develop a video game. There are a variety of game engines that can be used to develop 2D and 3D and each of them have their special features. After balancing all the features in each one of the game engines we decided to use the Unity3D game engine that is a powerful rendering engine fully integrated with a complete set of tools and rapid workflows to create interactive 3D and 2D content and has C# and JavaScript as primary scripting languages [20]. One aspect that pops up in sight when you use c# and JavaScript is the semantic similarity between the two languages which facilitates the developers work. Unity3D has a very simple, uncluttered interface for development that allows us to develop games quickly and has a very good integration with Microsoft Kinect sensor. It is also a game engine quite well documented with a strong Asset Store where you can download scripts, objects, sounds and textures that can be used to create a video game. It has amazing third party solutions for Audio and Physics and the code is well architectured to reduce the amount of errors done by programmers. Those were the main arguments that led us to select Unity3D as our game basis.

Unity3D game development interface is well structured and can be personalized and organized to the developers will, as seen in Figure 3. The interface has a scene creator where the developer can interact with the objects by dragging them, scaling, rotating, changing their disposition, format or texture (1). It has an area with a run time preview of the game where it's possible to see the impact of the changes made in real time (2), a document tree section where it's possible to see the folders structure and organization inside the project (3), an area to see the files contained in those folders and an area where we can visualize the properties of the scripts (4) and a console area that can be used for debugging purposes (5).

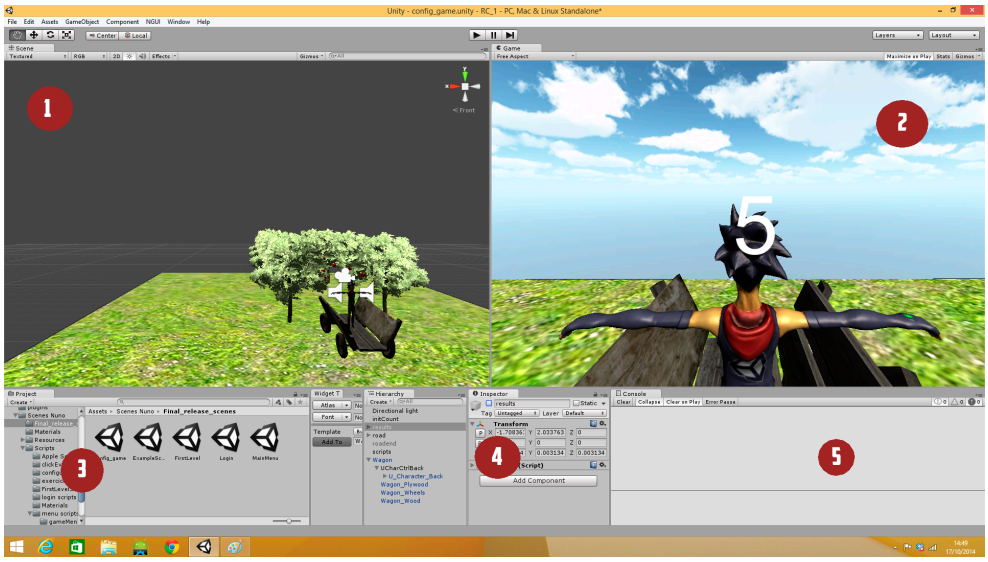


Figure 3 – Game engine work environment

One of the main important aspects of this game engine lies in the interaction between the objects in a scene. Unity3D projects have no main script that controls all game aspects, probably exists something that resembles to a main script, but Unity3D manages it and the developer has no need of interact with it. Scripts are seen as behaviours when attached to an object, so if we want an object to adopt a particular behaviour we just have to create a script extending a monobehaviour class, programme it and attached it to an object and the object will behaves as planed. The communication between the scenes objects can be done in two ways, the first one is by applying physics laws to the object, which means that if two object defined as rigidbodies and under physics laws touched each other the action-reaction principle will be applied to them and depending on the material they are made of they will adopt different behaviors. The other way to affect objects behaviour is by sending messages to it. An object can be programmed to send a message to another object in the same scene every time a specific event occurs. To send those messages we use the public methods that can be defined in the scripts that are attached to an object, and every time and object needs to communicate with another object it just have to find the object by its name or tag and send it a message using its public methods. This way objects can affect the behaviour of one or multiple objects at once, which makes the scene more real. This approach of splitting the code in shorter scripts is quite handy because it simplifies the code and helps in the organization of the project, which eases the debugging process and makes the code cleaner.

3.3 First tests

To be able to combine Unity3D game engine and Microsoft Kinect sensor, first we needed to understand how it works. We have used an asset available in the Unity3D store named “Kinect with MS-SDK” [21] that eases the integration process between Unity3D and Kinect.

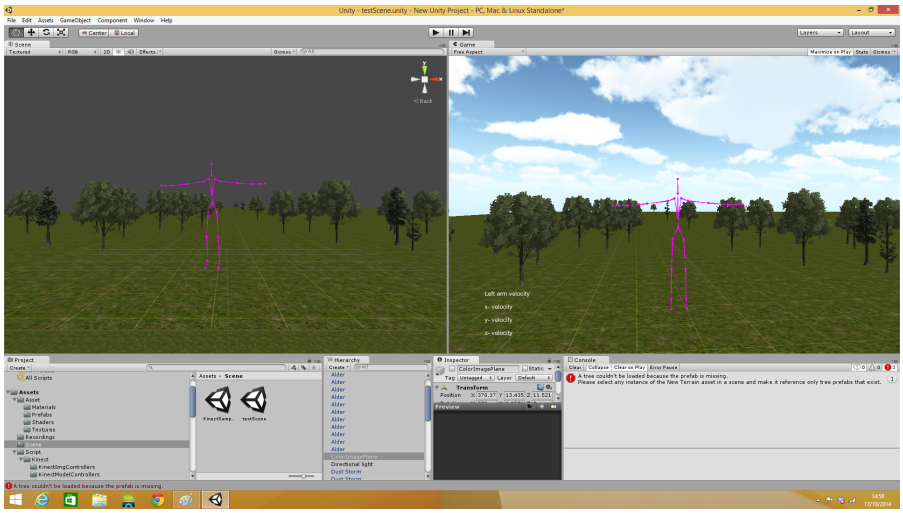


Figure 4 - First skeleton test and velocity measurements

Figure 4 illustrates the first test using Unity3D and Microsoft Kinect sensor. The main objectives were to test the Kinect sensor reaction to the movements performed by the user and how it reproduces the movement on the screen and analyse the output of the joints position in the console area. After analysing the output we start making tests and developing the angle calculating function, explained in section 3.6, based on those output values.

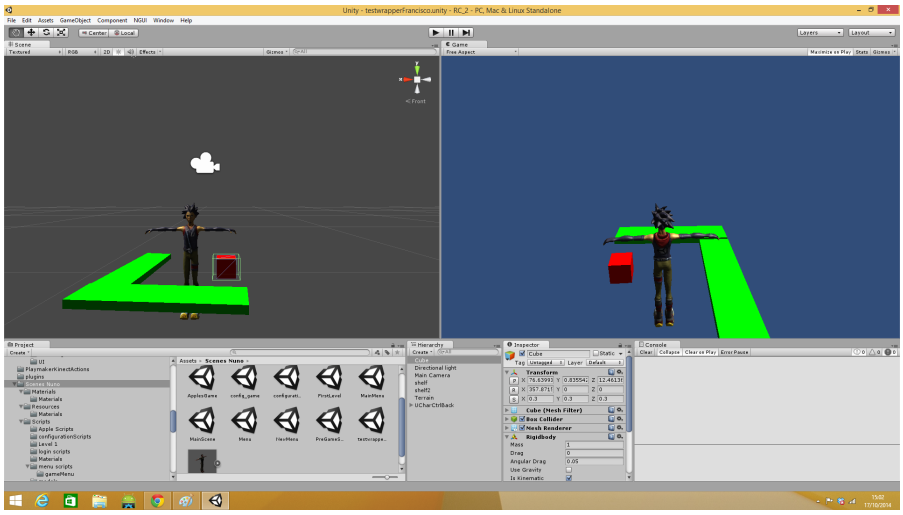


Figure 5 - Unity3D Collisions and messages first test

To be able to understand Unity3D interaction between objects we created this test, as seen in Figure 5, where the user must catch a red rigid box liable to physics laws and put it under a shelf. The main idea was to test object's collisions and implement messages. This way when the user catches the box, the box will send a message to the user's "hand" containing the "box" identifier, this allows the "hand" to attach the "box" to him using the identifier, this way when the user moves is hand the "box" will

move too. When the user lays the “box” on the “shelf”, the “box” will send the same message to the “shelf” which in turn will send a message to the “hand” saying that the “box” can be dropped and detached from it.

3.4 The Game Environment

The game environment should be clean, organized and reproduce a real life situation, but at the same time, allow patients to perform simpler and easy movements, which they can relate and carry to their daily tasks. Besides that some properties of the scene have to be configurable.

The chosen game environment is based on an orchard and intends to simulate a harvesting of apples. The harvesting theme as game base environment seems to be a really good solution and incorporates all the aspects referred above. It reproduces a clear and organized real life situation where patients can perform a set of movements that they can carry to their daily tasks. Due to the fact that the game pretends to simulate a harvesting, it is focused on the upper limbs of the patient. The game is divided in two levels, both configurable by the therapist, but with different goals.

3.4.1 First Level

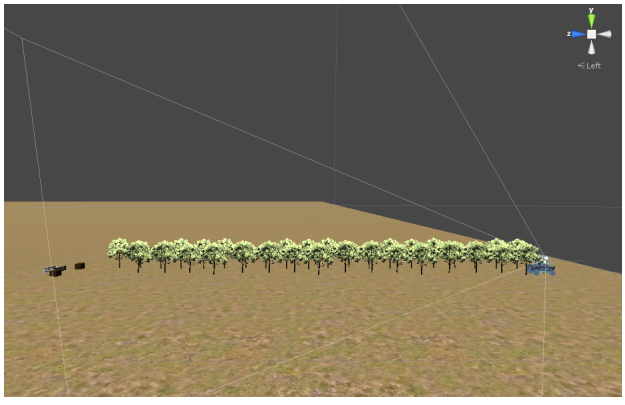
The fists level is a level where the patient can practice the movements repeatedly until he/she can perform the movement correctly. Level one should be used in patients as the fist part of the rehabilitation process, when the patient must understand and memorise the movement. This is achieved by making patients to perform repeatedly the same movement until they can perform it in a full range and with no pain. For that purpose the level is based on repetitions of the chosen movement. When programming the rehabilitation session the therapist choose the limb or limbs that must be trained, select the number os repetitions that should be done by the patient during the rehabilitation session and the angle that must be performed to achieve the session goal. When performing the rehabilitation session the patients are standing in front of the screen and receive indications about the movement that they should perform, as seen in Figure 6. Every time the patient executes a set of five repetitions correctly, an encouraging message appears on the screen. With these messages we intend to motivate the patient to continue executing the repetitions and achieve the session’s goal.

Figure 6 – First Level

3.4.2 Second level

The second level is more like a challenge level, where the main purpose is to work the patient's coordination by executing multiple movements simultaneously and combine that with his/her cognitive abilities. The scene is composed by rows of trees on both sides intending to simulate a harvesting of apples where patients must catch as many apples as possible for a defined period of time.

Due to the complexity of the level, it suffers significant changes during the development process. At fist we start by creating the entire scenario as seen in Figure 7.a, defining an endpoint, which means that the entire scenario was static, from the length of the terrain to the number of threes and the game stopped when the car reached the defined endpoint. This approach is very limited because one of the configurable parameters is the session time, and the game should continue to run until the time set for the session ends. Other reason against this approach is that it is computationally heavy because the entire scenario is rendered when the game starts. To be able to solve the problem we choose a more dynamic approach where all the objects in the scenario are rendered as we progress in the scenario, as seen in Figure 7.b.



(a) First approach

(b) Final approach

Figure 7 - Second level evolution

3.4.3 Points system

There are two different kinds of apples, the red ones and the green ones. Different kinds of objects to choose from allows us to define different weights for those objects, which allows the therapist to understand the cognitive capacities and the patient behaviour in situations where he has to take decisions. In this case, the red apples add 100 points to the game score and the green ones add 50 points, as seen in Figure 8, so the patient has to decide if he prefers to catch the green apples that are easier to catch and get only 50 point or if he prefers to catch the red one and reach the goal defined by the therapist faster. Other characteristic of this level is that it introduces movement, where the patient stands up in a trailer and the trailer moves in straight line through the orchard. The speed at which it moves is configurable by the therapist and can be increased as the patient makes progresses.

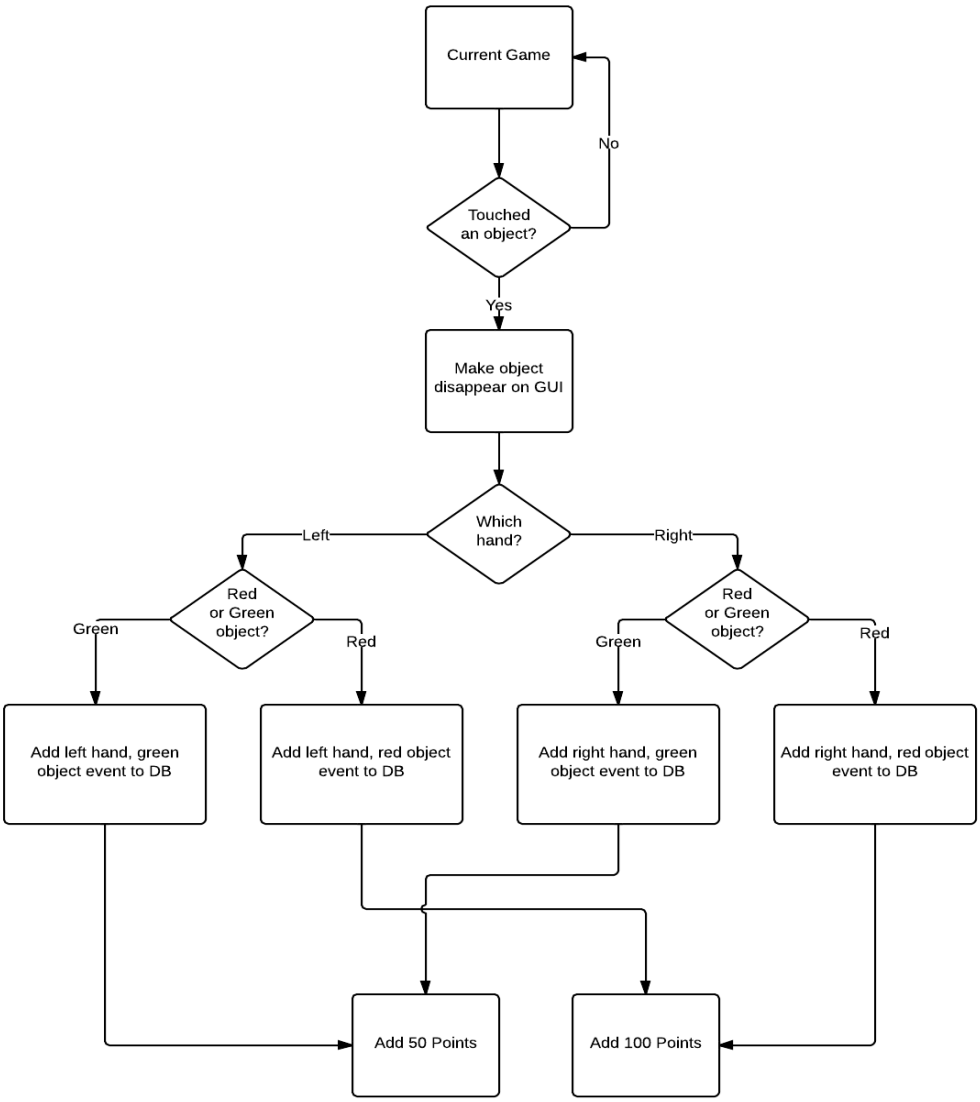


Figure 8 - Pickup objects flowchart

The main difference between this game and the other games mentioned in Related Work chapter is that the parameters related to the rehabilitation process are configurable by the therapist. Using the mobile interface (Chapter 6) the therapist can define rehabilitation parameters such as speed, session duration, if the user should or not execute rotation of the body, the difficulty level of the game given the patient's condition and the minimum number of points to do at a session setting up a goal for the patient to commit with.

3.5 Configuration features

One of the main objectives of the project was to develop a configurable game, which takes into account the needs of the patients so they can feel that they are practicing exercises that can be useful to them in daily tasks. The game has a set of configurable features; some of them are common to both levels others are specific to each on of the levels. All the configurations are made using the mobile application developed to assist therapists in rehabilitation session's management.

For the first level, based on repetitions, the therapist is able to choose the limb or limbs that the patient should use to perform the session, select a movement between the available ones, set up an angle that the patient must perform with the chosen limb then select the number of repetitions and the duration of the session. For the second level the therapist can select which limb should the patient use, the points goal, if the user must perform a middle point rotation or not, and the session time. The configurations are stored in the server and associated with a rehabilitation plan which is associated with user profile as described in the database diagram (Chapter 4).

To be able to access to a plan configuration, patients must login into the system using their credentials (e.g. Username, Password). If the user is correctly authenticated the database will return an identifier (id) of that patient. That identifier will be used by the system to request information about the patient, which means that it can be used to request the patient's plans and the plan's configuration. After load the configuration, the game will be set using those configured parameters and a preview of the goal of the game is shown to the patient as a tutorial as explain in Figure 9, using a flowchart.

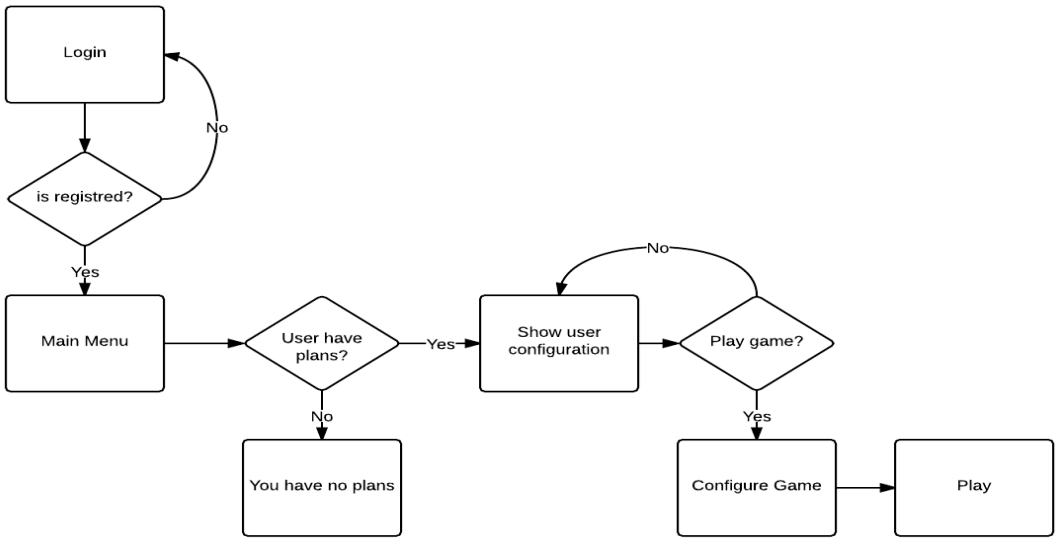


Figure 9 - Load configurations flowchart

Figure 10 shows the messages exchanged between the game application and the database to get the plan configuration. An API (Chapter 5) has been created to handle the communication. It behaves as an interface between the data in the database and the other two components of the system. This API deals with the requests from the game application and mobile application receiving the requests and processing them, returning a structured response that the other components can easily understand.

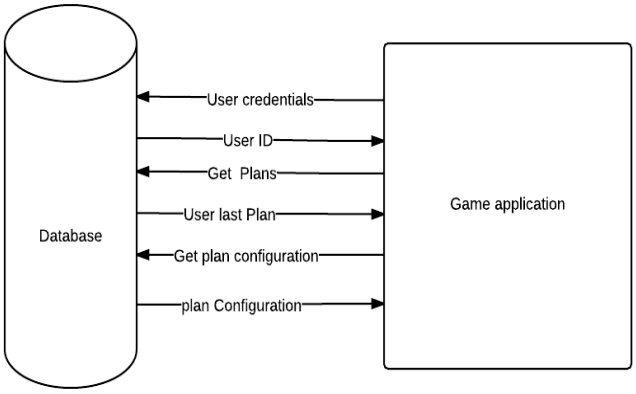


Figure 10 - Load configurations network messages

3.6 Data collection

To be able to create an Electronic Health Record for physiotherapy purposes, data has to be collected from the rehabilitation sessions of each patient and stored by data so it can be easily acceded and shared. The Kinect sensor is a powerful tool that detects 20th patient's joints and collects information about the position of those joints. The sensor collects information about the patient's body every frame like a screenshot, as seen in Figure 11, and transforms the detected patient's joints in three-

dimensional coordinates (x, y, z) so that we can work with them. Using the values collected by the sensor it is possible to calculate velocities, ranges, rotations and angles.

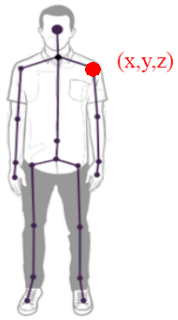


Figure 11 - Kinect Sensor skeleton joints

Since the Kinect sensor has a velocity of 30 FPS it would be a lot of information to process, so we decided that we should have a sampling frequency of one second, which means that we store the positions of the patient's joints every second. To that set of data collected in each second we called scene. Considering that velocity is given by displacement in function of time we can use the position of the joints in the previous scene and position of the same joint in the actual scene to calculate velocity. Since the Kinect sensor represents the position of each joint with a three-dimensional vector we can split the movement and calculate the velocity in the X-axis, Y-axis and Z-axis, as seen in equation 1.

To calculate the velocity of the X-axis, Y-axis and Z-axis we have to apply the equation of the velocity for each one of the three-dimensional joints positions:

$$v_x = \frac{\Delta x}{\Delta t} \Leftrightarrow v_x = \frac{x_{t2} - x_{t1}}{t_2 - t_1}$$

$$v_y = \frac{\Delta y}{\Delta t} \Leftrightarrow v_y = \frac{y_{t2} - y_{t1}}{t_2 - t_1} \qquad (1)$$

$$v_z = \frac{\Delta z}{\Delta t} \Leftrightarrow v_z = \frac{z_{t2} - z_{t1}}{t_2 - t_1}$$

We can also calculate the average velocity for each axis using a sampling window of N samples:

$$v_{x_{av}} = \frac{1}{N} \sum_{n=0}^{N} \frac{x_{n+1} - x_n}{t_{n+1} - t_n}$$

$$v_{y_{av}} = \frac{1}{N}\sum_{n=0}^{N}\frac{y_{n+1}-y_n}{t_{n+1}-t_n} \qquad (2)$$

$$v_{z_{av}} = \frac{1}{N}\sum_{n=0}^{N}\frac{z_{n+1}-z_n}{t_{n+1}-t_n}$$

To calculate the angles we created a C# class that we called JointsHandler (Appendix A). This class is an interface between the joint positions and the rest of the game classes. When we instantiate an object of this type, this object automatically manages the patient's joints data and we can ask it for the position of a specific joint or to calculate an angle. To calculate an angle we used a function, as seen in Figure 12, which is based on equation 3, that receives three tree-dimensional vectors and calculates the angle of the one in the middle.

Calculating the angle between three three-dimensional vectors using a dot product property:

$$\overrightarrow{AB}\cdot\overrightarrow{BC} = \left\|\overrightarrow{AB}\right\|\left\|\overrightarrow{BC}\right\|\cos\theta \Leftrightarrow \theta = \cos^{-1}\left(\frac{\overrightarrow{AB}\cdot\overrightarrow{BC}}{\left\|\overrightarrow{AB}\right\|\left\|\overrightarrow{BC}\right\|}\right) \qquad (3)$$

Implementation of the dot product:

```
17    // calculate the angle based on joints positions
18    public static double AngleBetweenJoints(Vector3 j1, Vector3 j2, Vector3 j3)
19    {
20        double Angulo = 0;
21        double shrhX = j1.x - j2.x;
22        double shrhY = j1.y- j2.y;
23        double shrhZ = j1.z - j2.z;
24        double hsl = vectorNorm(shrhX, shrhY, shrhZ);
25        double unrhX = j3.x - j2.x;
26        double unrhY = j3.y - j2.y;
27        double unrhZ =j3.z - j2.z;
28        double hul = vectorNorm(unrhX, unrhY, unrhZ);
29        double mhshu = shrhX * unrhX + shrhY * unrhY + shrhZ * unrhZ;
30        double x = mhshu / (hul * hsl);
31        if (x != Double.NaN){
32            if (-1 <= x && x <= 1)
33            {
34                double angleRad = Math.Acos(x);
35                Angulo = angleRad *(180.0 / Math.PI);
36            }
37            else
38                Angulo = 0;
39        }
40        else
41            Angulo = 0;
42
43
44        return Angulo;
45
46    }
47
48
49    private static double vectorNorm(double x, double y, double z){
50        return Math.Sqrt (Math.Pow (x, 2) + Math.Pow (y, 2) + Math.Pow (z, 2));
51    }
```

Figure 12 - Angle calculator function

As example, if we want to calculate the angle of the right shoulder the arguments of the function would be, the position of the right hip, the position of the right shoulder and the position of the right elbow

respectively and the output will be and angle in degrees of the shoulder. Other particularity of this class is that it is an interface with the database, dealing with the requests to create a new scene and store it in the database.

3.7 Game Feedback

One of the most important components that a game should have to be able to stimulate the patient to continue playing the game is feedback. The game has to communicate with a patient to create engagement. Every time the patient accomplishes a goal, should be congratulated and encouraged to continue doing the exercise correctly. For that purpose we introduced audio and visual feedback. The audio feedback is really useful to indicate if the patient has touched an object and to stimulate the patient's senses so, every time a patient touches an object from the scene he/she can hear a feedback sound. In regards to visual feedback, it was added into the game using two different types of feedback. The first one is used simultaneously with the audio feedback, it appears every time the patient pickup an object and consists in a visual feedback indicating the number of points gain by the patient for catching that object, as seen in Figure 13. The second one consists in encouragement phrases, similar to a popup and they appear randomly on the screen every time the patient accomplishes something, e.g., if a user executes a set of repetitions correctly a popup message will appear to congratulate the user of his accomplishment, as seen in Figure 14.

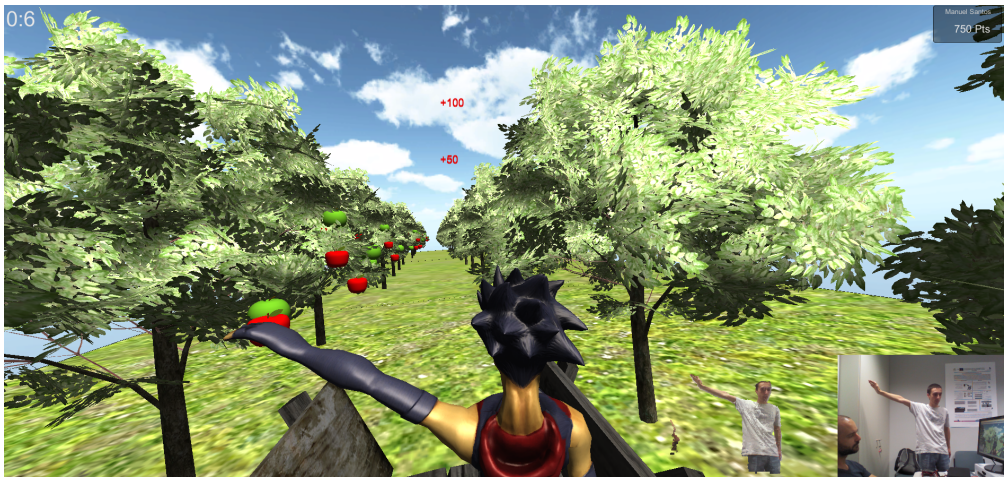


Figure 13 - Points visual feedback

Figure 14 - Popup messages visual feedback

3.8 Conclusions

Video games play a very important role in rehabilitation sessions. They are able to increase the patient's engagement, which translates in a faster recovery. These games called "serious games" have an active and important role in physical rehabilitation future. To be able to perform that role, these games must be configurable and capable of collecting data from a patient rehabilitation sessions. The data will be used to help therapists analyze the patient progression and to create an EHR to physiotherapy. To achieve that we use a motion-sensing devise called Kinect, which can detect patient's body position and replace the remote control for it. By combining a video game development platform called Unity3D with the Microsoft Kinect sensor we were able to develop a 3D video game with those characteristics.

4 Database

A database is a collection o data that are typically organized to model aspects of reality in a way that supports processes requiring information. When developing applications that have to deal with a huge amount of data, a well-structured and well-optimized database is one of the most important aspects to have in mind. A well-structured database is one that has well defined objects and well-defined relationships between those objects, which eases the access.

There are a lot of different types os databases, analytic databases that are read-only databases which store archives, historical data used for analysis, operational databases that are used to manage more dynamic bits of data, model databases that diferenciastes databases according to how they model the data, hierarchical databases that define hierarchical-arranged data and network databases that were designed to solve some of the more serious problems of the hierarchical databases as data redundancy [22].

4.1 Management and Administrations Tools

To create and develop a database we used a database management system (DBMS), which is a specially designed software application that interacts with the user, other applications, and the database itself to capture and analyze data. The main purpose of a DBMS is to allow the definition, creation, querying, update, and administration of databases. We decided to use a database management system based on the relational model (RDBMS) named MySQL [23], which is one of the most used open-source DBMS and combined with PHP it is cross-platform which means that we can develop in Windows or Unix. MySQL uses SQL, which is a special-purpose programming language design for managing data held in a RDBMS.

MySQL has no GUI tools to administer databases. For that reason, a free integrated front-end tool named MySQL Workbench [24] that enables users to graphically administer MySQL databases and visually design database structures was used. This visual tool enables developers to visually design, model, generate and manage databases. It allows developers to do forward and reverse engineering on database allowing them to export a graphically designed database model to a SQL file, which can be used to replicate the database everywhere. Data can also be exported. Though the Database Connections Panel, developers can easily create, organize and manage database connections, which means that it allows multiple connections to databases held on servers, and to perform real-time changes. Other tool provided by Workbench that is really useful is the SQL Editor. It provides color syntax highlighting, auto-complete, reuse of SQL snippets, allows the execution of multiple queries simultaneously and the visualization of the results in separated tabs. A Table Data Search Panel that allows us to export data to common formats as CSV, HTML or XML is also provided.

4.2 Database structure

We can divide de data stored in the database in two main data types. The first type is the data related to users, as seen in Figure 15, that is usually added to the database through the mobile application interface: patient's personal data, notes written by the therapist, prescribed plans, configurations, therapy history, medication history and clinical history. The second type is the data collected by the Kinect Sensor, regarding to the game sessions, as seen in Figure 16.

4.2.1 User's stored data

To store users we had to define two profiles with different permissions levels in the system. The fist profile is the therapists profile, which means that the user has high permissions and can access and change patient's information if that patient is defined as being under his supervision. Therapists can only access to patient's information under their supervision because most of the time patient's want to have professional discretion. To access and manage that information therapist have to use the mobile interface. The second profile is the patient's profile that has low permissions and it can only be used to login in the game and access to plans configuration. If the user is associated with a patient profile, he will have his clinical, medications and therapy history associated to his profile as well as notes related to previous performed sessions and prescripted plans. Each patient can have a set of plans associated to his profile that are defined by his therapist. All plans have an associated configuration defined by the therapist during the creation of the plan, which will be used in the construction of the 3D game scene when the user starts a session in the game.

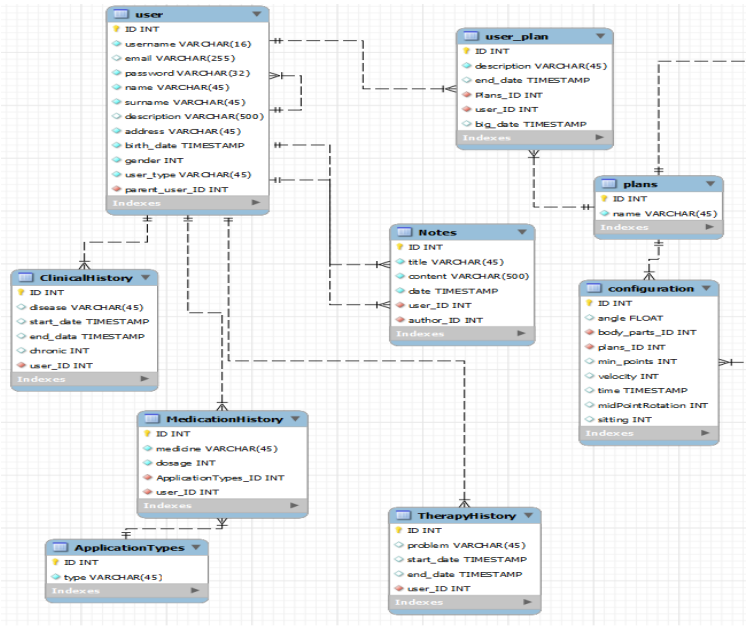


Figure 15 - User´s data model and relationships

4.2.2 Game stored data

Every time a patient starts a new game using the game application, it will connect to the database logging in the patient in the system and creating a new session associated to the plan that was previously created by the therapist. This way we can create a history of past therapy sessions organized in time. Every session has an ID that is used to identify the session. When the server receives a request from the game application to create a new session for a specific patient, it will answer the request sending that identifier if the operation is well succeeded. During that session lifetime, every time the game application needs to perform a request to de API, it will use the session's ID to make it. This way all the data that regards to that session will be connected and organized in time.

As said in Chapter 3, we used a sampling frequency of one second to store the patient's body positions. To that group of three-dimensional vectors (x, y, z) corresponding to the position of the head, torso and upper limbs we called scene. Each scene is associated to session by its ID, which means that a session is a set of scenes organized in time. Each scene contains the X, Y and Z positions of a specific joint from the patient's body. A joint can represent a body part like a hand or in some cases a group of joints is needed to represent it a body part, as with the arms. To be able to make that representation we create a relation between joints by previously defining that an arm is represented by the hand, the elbow and the shoulder, so if we want to store the right arm position we just have to use the right arm identifier and the system will automatically store the positions of the joints that compose that arm. Storing these data by scenes organized in time we can replicate all the patient's movements during a session, as seen in Figure 17. This is very useful because despite all stats provided to the therapists by the mobile application, some times they like to analyze the patient limbs position. Initially we thought in taking a picture every second, but due to privacy issues we consider that this alternative was less invasive. In addition to the data collected by the sensor, there are special events like picking up an apple, that are saved so that we can map the evolution of the patient, by comparing the number of picked up apples between a set of sessions performed by that patient. If the number of picked up apples increase, that could mean that the patient is improving.

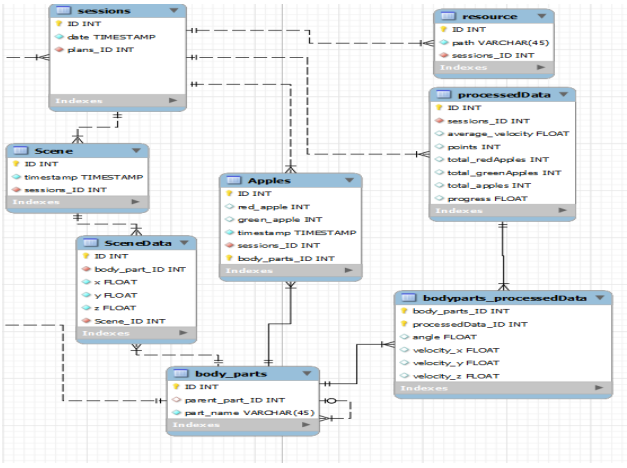


Figure 16 - Game data model and relationships

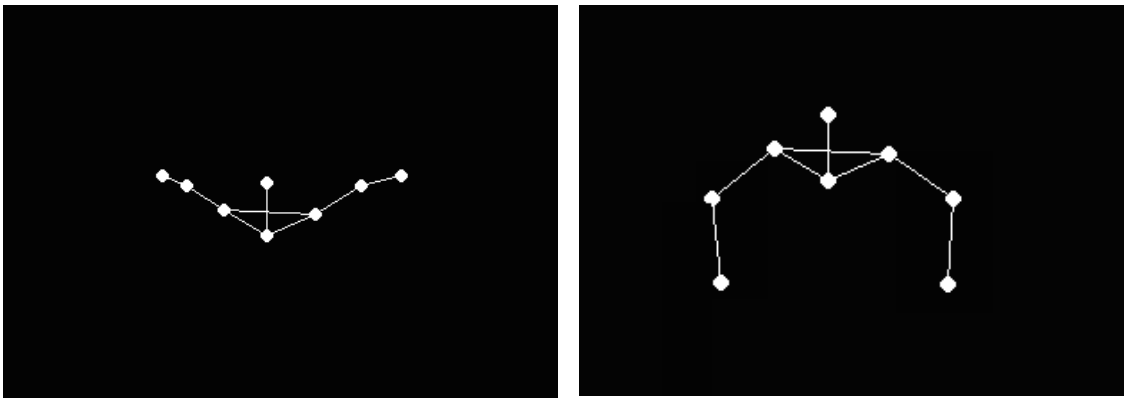


Figure 17 - Patient's position reconstruction

Figure 17 illustrates the reconstruction of a patient position using the three-dimensional coordinates. Each one of the images reproduces the position of the patient's body joints in a scene, showing that storing consecutive scenes, we can help us to reproduce all the movements performed by a patient during a rehabilitation session.

4.3 Conclusions

To support a data collecting system able to store sufficient information to create an EHR a structured and organized database is needed. In this chapter the approach and implementation of a database that can perform this work is discussed. Tools like MySQL and MySQL Workbench used to develop and manage the database were described and explained. Regarding to the database organization, the approach of having two distinct categories of data stored in our database, user data and game data, was explained as well as the relationship between those categories objects. The strategy adapted to store and organize the three-dimensional vectors corresponding to the patient's joints that are collected by the Kinect sensor is also described.

5 API

An Application Programming Interface (API) [25] is an essential tool for high quality system architectures. It allows one component of the system to make use of the functionality of, or data available to, another, by providing a consistent, programmatic method for accessing a resource. It is simply a structured way of exposing functionality. APIs make it easy to efficiently share data and processes. They are an ideal tool for increasing awareness of work done and services offered, in a different manner. APIs provide interoperability and offer a data infrastructure on which other can build. A successful implementation of an API results in a greater flexibility and improves the ability to efficiently present outputs as useful and re-usable artefacts.

To be able to create an application capable of storing data and complement the current EHR system was considered the development of an API-centred system. By creating a system in this manner, it eases the connection and creates interaction between system components.

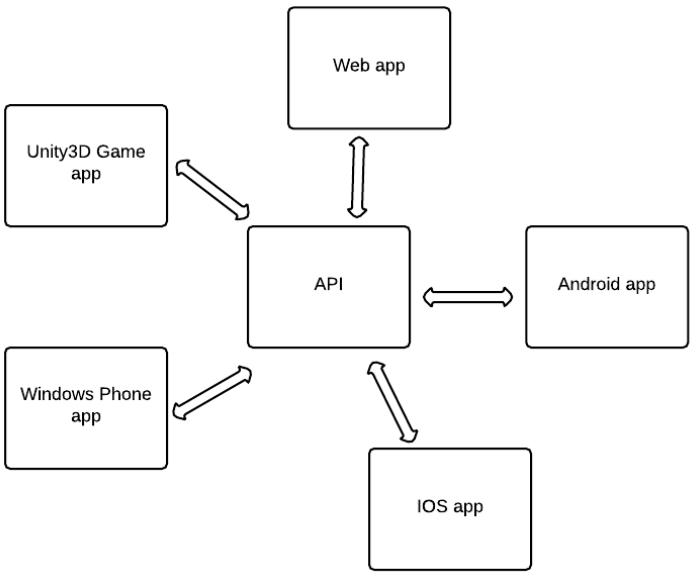


Figure 18 - API-Centred system

Figure 18 illustrates the usefulness of an API-based system. Using an API we standardized the requests and responses, creating a service, which means that regardless the develop application language or platform as long as they are able to execute HTTP requests and read JSON objects **[26]**, they can use the service and have access to the data stored.

5.1 Implementation of the API

To implement the API we used PHP [27], which is an open-source language, easy to use, well supported by a large group of developers and well documented, relatively fast, has a powerful support library and has a good and easy integration with MySQL databases.

To access MySQL databases we used an access layer called PDO_MYSQL, which is a driver that implements PHP Data Objects (PDO) to enable access from PHP to MySQL. The PHP Data Objects extension defines a slight and consistent interface for accessing databases in PHP. It does not provide database abstraction by rewriting the SQL queries or emulating missing features, it provides data-access abstraction, which means that, you can use the same functions to fetch data regardless the database.

5.1.1 API Structure

The structure of the API has a core script called kinect.php, which is the script that creates and manages the connection with the database. It receives all the requests form the other components of the system, game applications and mobile applications, process those requests and begins the response process. To ease the development process and later changes, we divided the API in six different folders each one containing a script, as explained in Appendix B. Each script aggregates all the methods needed to deal with a specific type of data. Those main scripts are stored in separated folders each one with a name referring to the script they contain. The user.php script contains all the method needed to deal with users (e.g. Add, List, Delete), the notes.php deals with the notes written by therapists, plans.php deals with the plans and also with the configuration of those plans, the history.php grant access to the patient's medical EHR allowing therapists to consult or add items to it, the login.php script deals with the login process verifying the user credentials and allowing access to data and the sessions.php which is the most extensive and complex script deals with the game data, allowing the creation of therapy sessions, games scenes and to store the patient's joint positions.

```
// lists all the users that has the type patient for a specific physiotherapist  !!!!!!!!!!!!!! should return the element id !!!!!!!!!!!!
function listPatients($parent_ID){
    global $dbh;
    $sth = $dbh->prepare('SELECT name, surname, ID  FROM user WHERE parent_user_ID = :id');

    $sth->bindParam(':id' , $parent_ID);

    try{
        $sth->execute();
    }catch(Exception $e) {
        echo '<h1>An error has ocurred.</h1><pre>', $e->getMessage() ,'</pre>';
    }

    $result = $sth->fetchAll();
    $array_to_send = array();
    $arr = array();

    // enconde result var as a json object for comunication between android and server  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    if(count($result) > 0) {
        foreach($result as $r) {

            array_push($arr, array('name'=>$r['name'], 'surname'=>$r['surname'],'id'=>$r['ID']));

        }
    }
    //var_dump($array_to_send);

    //$array_to_send['users'] =array();
    $array_to_send = array('users'=> $arr);
    echo json_encode($array_to_send);


}
```

Figure 19 – List patients function

Figure 19 illustrates a function contained in the user.php script that is used to list the patients that are associated with a therapist. The function receives as a parameter the ID of the therapist and returns a JSON array containing the name, surname and ID of the patients.

5.1.2 Connecting to database

Different databases may have slightly different connection methods, but using PDO they are all almost similar, which eases the connecting process. As seen in Figure 20, the resemblances between a connection to a MySQL database and two other database types are clear. PDO intend to make this type of connection as similar as possible.

To connect to a database it is only necessary to identify the type of the database along with the host address, and then indicate the database name, the username of the user that has permissions to access the database and the corresponding user password. DBH stands for 'database handle' and it must be defined as a global variable so you can access it in other scripts, this variable is the one that detains the database connection and every time we need to perform a database operation it must be used.

```
try {
      # MS SQL Server and Sybase with PDO_DBLIB

      $DBH = new PDO("mssql:host=$host;dbname=$dbname, $user, $pass");
      $DBH = new PDO("sybase:host=$host;dbname=$dbname, $user, $pass");

      # MySQL with PDO_MYSQL

      $DBH = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
}
catch(PDOException $e) {

      echo $e->getMessage();
}
```

Figure 20 - Perform database connection using PDO in different database types

5.1.3 Handle a Request

To perform a request the client must define the type of action we wants to make, e.g. list a therapist's patients, after that the therapist has to define the parameters needed to execute that action, as seen in Figure 21. The request is always sent to the kinect.php script, which verifies if the action exists and if the parameters received match with the ones needed to execute that action. After making that verification it will call the script containing the method that is defined to deal with that type of request, passing it the parameters received. Figure 22 illustrates how the system handles with requests through a sequence diagram.

```
WEBROOT_PATH/kinect.php?action=listpatients&pid=<parentID>
```

Figure 21 - Example of a request using the API

When a method is called to deal with a request, it must analyse the content of each parameter received to prevent it from being void. After that verification an SQL query must be created to perform the action in the database. For that, we used prepared statements, which are precompile SQL statements that can be executed multiple times by sending just the data to the server. It has the added advantage of automatically making the data used in the placeholders safe from SQL injection attacks..

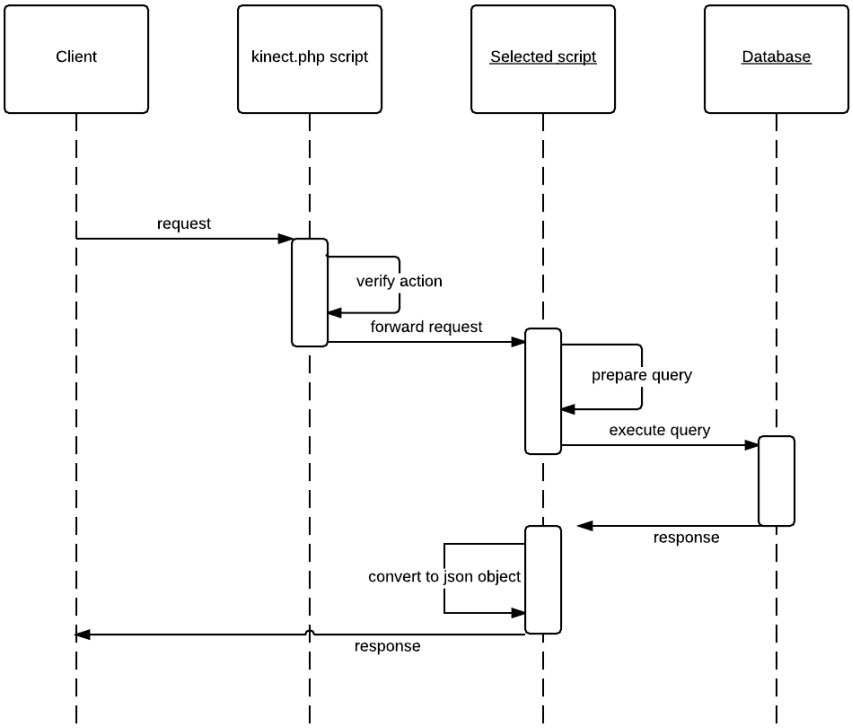


Figure 22 - Request sequence diagram

5.1.4 Response

After receiving the data form the database, the API has to present data to the client in a simple and organized way. For that reason we decided to represent the data using a language-independent data format named JSON, which is lightweight data-interchange format easy for machines to parse and generate. A JSON data object is a collection of name/value pairs and the code for parsing and

generates those objects is available in a large variety of programming languages. Using the above example of getting the therapist's patient list, the structure of the response sent to the client application using JSON is illustrated in Figure 23.

```
{"users":[
        {
        "name":"Manuel",
        "surname":"Santos",
        "id":"7"
        },
        {
        "name":"Nuno",
        "surname":"Duarte",
        "id":"8"
        }
]}
```

Figure 23 - JSON response structure

5.2 Conclusions

In this chapter we discussed the importance of having an API-centered system to be able to implement an EHR in physiotherapy. It is clear that a system architecture based on an API provides interoperability and offers an important data infrastructure, allowing us to develop innumerous client applications without changing the data access method. The structure of our API is explained as well as the reasons to use tools like PDO in the database connecting and management process. All the process of dealing with a request is explain as well as the response and the use of JSON providing efficient and structured outputs.

6 Mobile Application

The need of a mobile application as an interface between the data held in the server and the therapist is imperative. Mobile applications allow users to access data everywhere. This ability is a huge advantage to therapists, because most of them have to deal with more than one patient at once so it becomes practically impossible to monitoring all patient's sessions correctly and with proper attention. The novelty of the designed apps is the ability to configure the serious game according to the patient's needs as explain in section 6.3.3.

6.1 Structure

The mobile application was designed to materialize the interface between the physiotherapist and the system. The mobile application is only for therapists and after the login therapists can only see the list of patients that were assigned to them due to privacy issues. Figure 24 illustrates the structure of the mobile application by layers. After selecting a patient from the list the therapist can consult the patient profile where he/she can see the medical history of the patient, the plans assigned to that patient and the notes written. These tasks are secondary because one of the main purposes of the application is to access a patient session's data. Using the mobile application therapists can access data from the patient's previous therapy sessions, which is presented to the therapist using charts, as explain in section 6.3.1 and it also offers the possibility to compare the last five session of the patient to understand his/her progression. The charts show the progression of the patient in terms of points, and red or green apples. Regarding to a session data, it is possible to understand which hand the patient uses more when playing the apples game by comparing the number of apples caught using each hand, or specify if the patient catches more green or red apples using his/her right or left hand.

Configuration was the other main task that we wanted to integrate in the mobile application. Allow therapist to configure some parameters of the game helps the system to adapt the game according to the patient's needs, which creates engagement because patients feel that the game was build specifically for them. To allow configuration we needed an UI that was easy to use and to understand so we created a setup wizard that simplifies the process of configuration as explained in section 6.3.3. The setup wizard will allow therapist to configure a therapy session faster and without much effort.

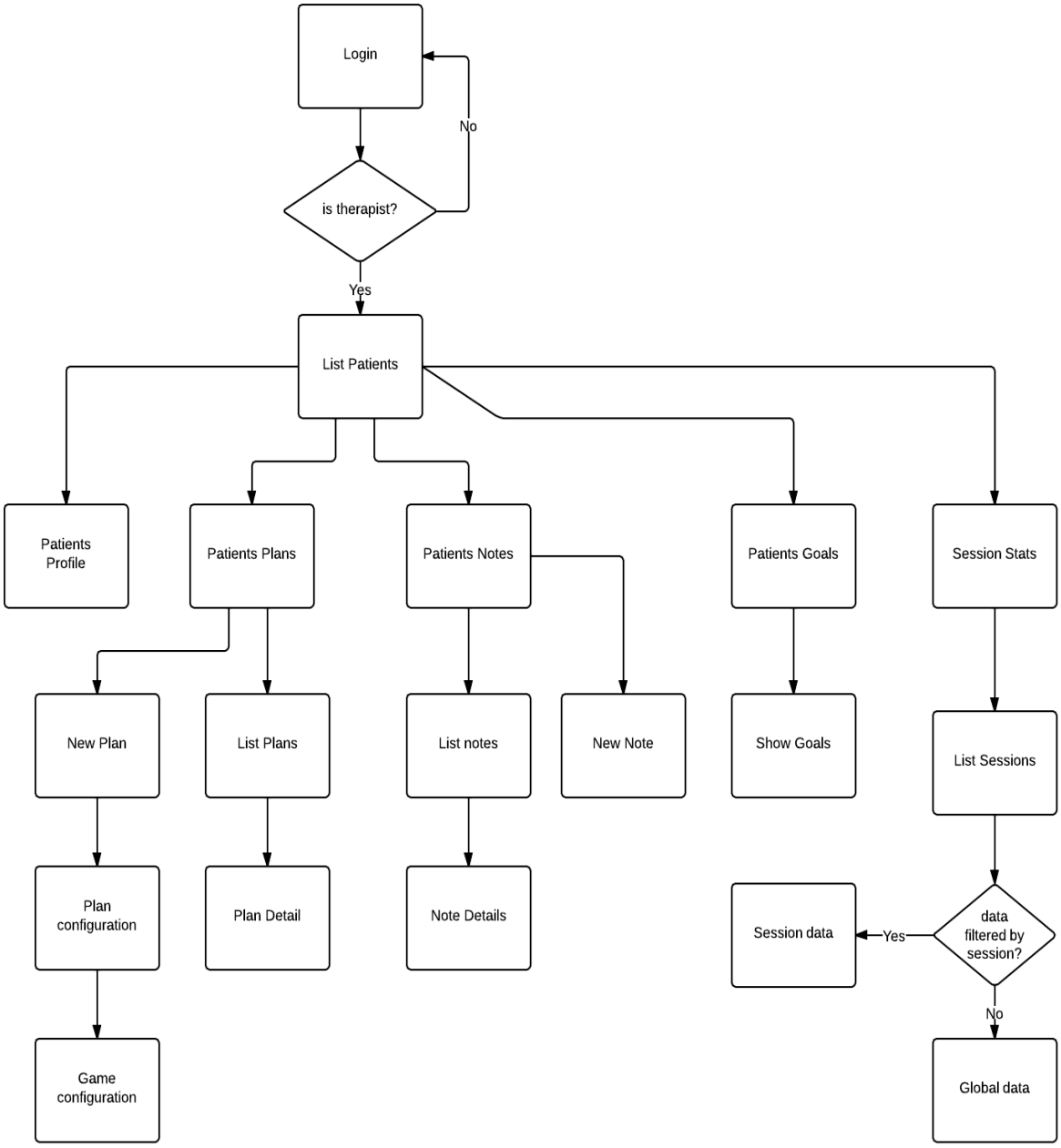


Figure 24 - Mobile Application flowchart

Regarding to the application structure, we follow the Android application development patterns that consists of top level and detail views. The idea was to create a hierarchical interface were the top level views display generic data and expose all the functionalities of the application where users can navigate from a view displaying generic data to a view displaying a more detailed data.

6.2 Client – Server Communication

The communication between the mobile application deployed in Android device and the server is done using a networking library named Volley [28]. Volley is an HTTP library that makes networking for Android applications easier and most importantly faster. Volley allows developers to automatically schedule network requests and handle multiple current network connections with minimum effort. It supports request prioritization, which means that developers can set different levels of priority (low, medium, high) to request data, allows request ordering making it easy to correctly populate the UI with data fetched asynchronously from the network. Besides that, volley has a cancellation request API that allows programmers to cancel a simple request or set blocks or scopes of requests to cancel. Regarding to the execution of the requests Volley eases those processes by reducing the code complexity of the requests.

Before Volley to execute an HTTP request we needed to create and launch a background task named AsyncTask and override some methods of that class as the doInBackground, where is possible to define the main action of the task, then we need to configure the onProgressUpdate that is used to display any form of progress in the user interface and the onPostExecute method that defines what should be done when a task ends. Because Volley is callback-based, it means that we just need to execute the request and define a callback function to deal with the response, which means that when the response arrives the function will be executed.

Other grate advantage of Volley is that it has transparent disk and memory caching with standard HTTP cache coherence often used in multiprocessors systems. Regarding to the responses, Volley deals really well with JSON responses so this was a strong argument to use JSON in our API responses. If the response is well structured we can even pass the JSON response directly to an Android list adapter and it will deal with it and populate that list correctly.

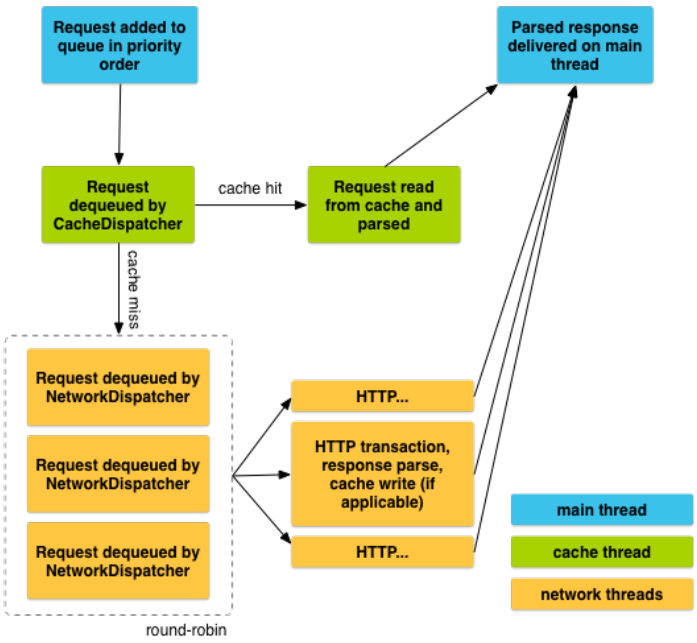


Figure 25 - Illustration of a Volley request life (Android Developers)

Figure 25 illustrates the life of a Volley request where it is possible to understand how it works. When a request is added to the queue, that request is picked up by the cache thread and triaged: if the cache can service that request, the cached response is parsed on the cache thread and the cache thread will deliver it to the main thread. If the cache cannot service the request, the request is placed on the network queue. When a network thread is available, it takes the request from the queue and performs an HTTP transaction, when the response arrives the network thread parse the response to the worker thread, writes the response to cache and posts the parsed response back to the main thread for delivery.

All the expensive operations like blocking I/O and parsing/decoding are done by worker threads. It is possible to add a request from any thread but the response will be always delivered on the main thread.

6.3 Graphical user interface

The interface is one of the most important aspects in a mobile application. An application without a clear and easy to use interface behaves as a barrier between the user and the data. The best way to create a simple and clean interface is following the patterns defined by Android for the creation and development of GUI [29]. These patterns define that the navigation of an application must be consistent to create a great user experience, the navigation between screens must be simple and the use of ListViews or GridViews in this process is suggested, which create organized content in clickable items that guide a user to a more detailed content. Regarding to the statistics of the session the use of chats seems to be the best way to present data and compare it in a clear way. Charts are a very efficient way to track progression and evolution or accentuate differences between data.

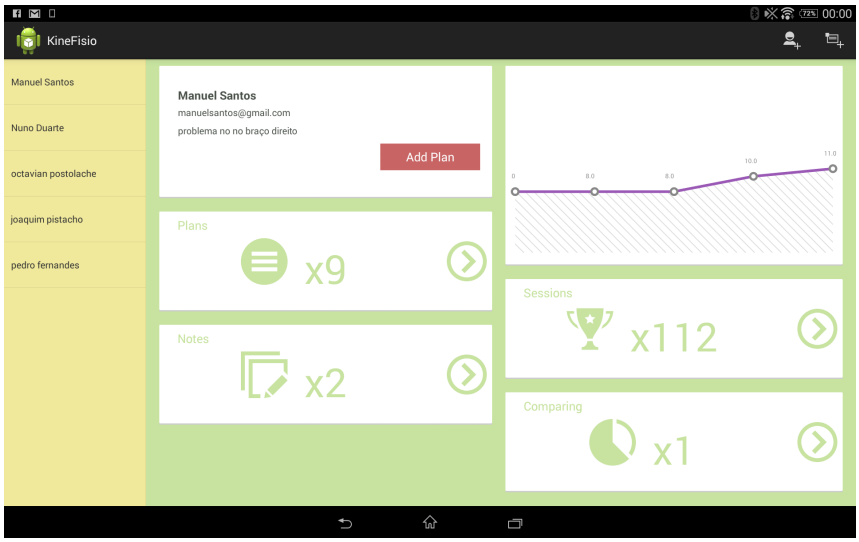


Figure 26 - Application main interface

Figure 26 illustrates the main interface of the application and it is a proper example of the implementation of the Android applications design patters. It is possible to see that this view embraces all the main features included in the application and give the user a preview of the content, by displaying a patients list and when a patient is selected the profile data organized in clickable items that lead us to a more detailed data is presented. A linear chart showing the patient's progression is also presented. All of those items show a preview of the number of elements contained within it. In the upper right corner it is possible to see two buttons, one for adding a new patient to the system and other for adding a new note.

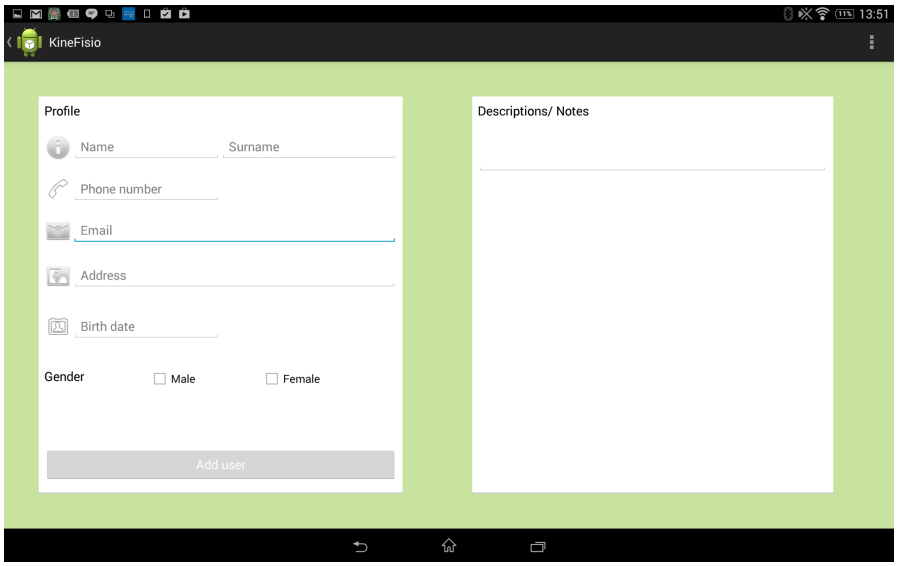


Figure 27 - Add user view

The process of analysing the data collected from a patient's session starts in the creation of the patient's profile. Figure 27 illustrates the interface used to add the patient's personal information to the database and create the patient's profile in the system. The therapist should introduce the patient's name, phone number, email, address, birthday date and gender as well as a description of the problem for which the patient will be treated. This process works as a registration process thereafter a password by default and a username will be created to allow the patient to login in the game application.

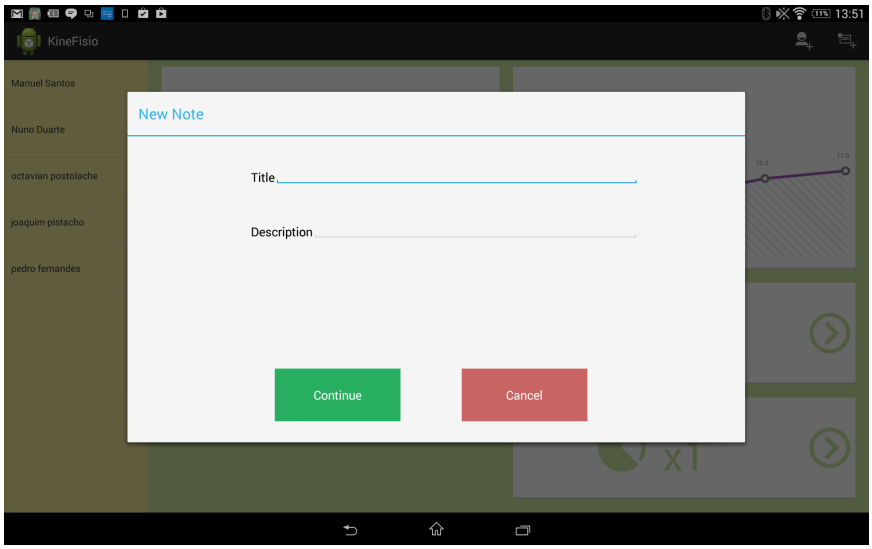
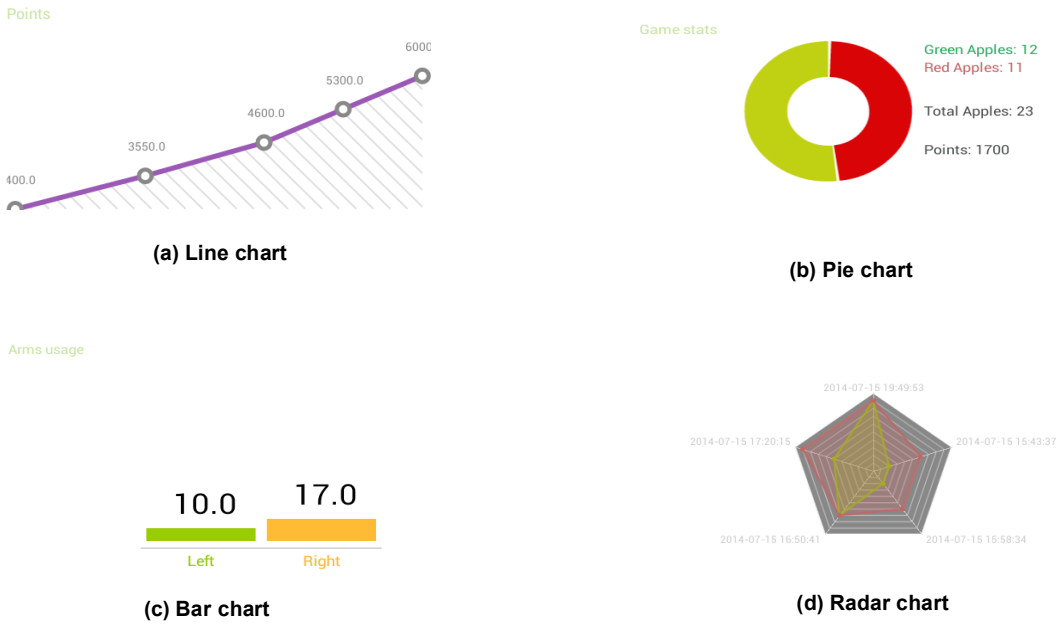


Figure 28 – Add new note

Regarding to notes, we created a dialog that allows therapists to take quick notes about the selected patient, as seen in Figure 28. The dialog is available from every section of the application and it overlaps to the content when the note button is pressed, so that the therapist doesn't have to stop what he/she is doing to take a note. A note consists in a title field and a description field and is automatically associated to the patient's profile that was previously selected.

6.3.1 Charts

It is important that the application can display the data in a clear, synthesized and objective way so that the therapist can easily understand them. To achieve that we use a set of different types of charts as line charts, pie charts, bar charts and radar charts to display the data graphically. To create the charts we library called HoloGraphLibrary [15] was used. It is simple to use and to integrate with the applications. It uses Android Canvas, which works as an interface to the actual surface upon which your graphics will be draw. Canvas is the best choice when your application needs to regularly re-draw itself. However, this library only has line, bar and pie charts so we had to implement the radar charts. This type of chart is really useful when comparing parameters of multiple sessions, as seen in Figure 29.d, which illustrates the amount of red and green apples picked up by a patient in the last five sessions. To implement the radar charts we followed the guidelines of the HolographicLibrary and use Android canvas. This implementation allows us to integrate the radar chart in the HolographicLibrary extending it, and homogenizes the chart construction process.



(a) Line chart

(b) Pie chart

(c) Bar chart

(d) Radar chart

Figure 29 – Mobile application charts examples

6.3.2 Statistics

The data collected from the game sessions has to be presented in a way that allows the therapist to easily analyse the data and draw conclusions. To achieve that, we implemented two modes of presenting data and create statistics.

The fist one is presenting the data from a session individually, as seen in Figure 30, which allows the therapist to see detailed data regarding to the patient's limbs. Using this mode the therapist can see a list of sessions executed by one patient and choose which one we wants to analyse. After choosing the session the data is presented graphically using charts, as noted in section 6.3.1. Using the data collected by the Kinect sensor and the methods of the developed API it is possible to calculate the success rate, which represents a ratio between the number of objects that can be picked up by the patient and the number of objects that he actually took, compare the limbs performance, understand which limb is more frequently used to pick up apples, the progression of each member, with which hand the patient does more points and with which hand the patient catches more green or red apples. All of these data allow the therapist to analyse the patient's limb behaviour and the cognitive capacities during a single session.

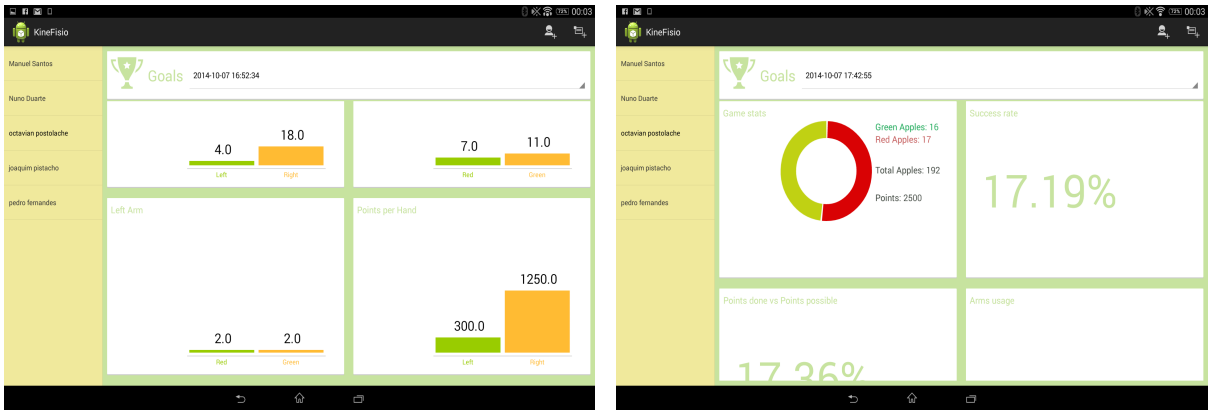


Figure 30 - Session detail data view

The other mode allows the therapist to compare sessions, which is important to understand and track the progression of the patient during the rehabilitation process. When choosing this mode, charts showing the progression in game parameters like the success rate, number of points, the number of green and red apples caught in each session is shown. These parameters allow the therapist to understand the physical and cognitive evolution of the patient.

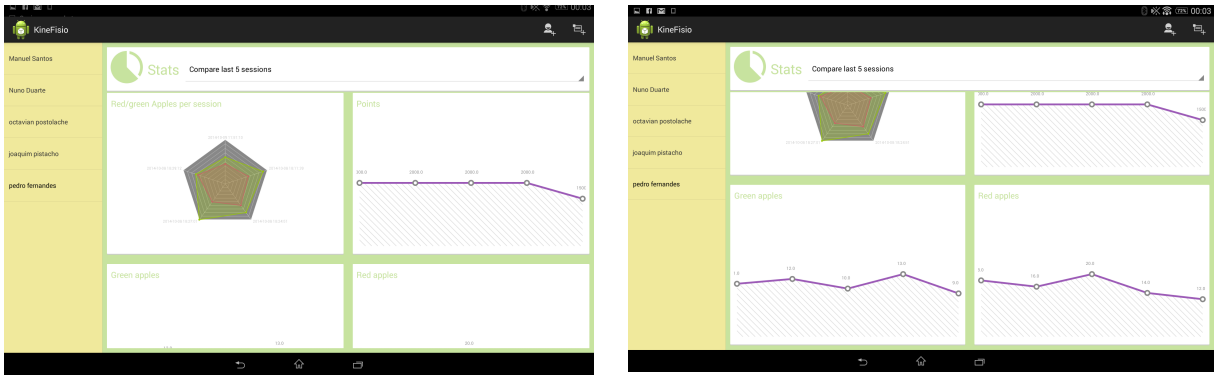


Figure 31 - Compare last 5 sessions view

In Figure 31 a comparison of the performance of a patient in the last five sessions is made. It is possible to see a radar chart comparing the number of red and green apples picked up, line charts illustrating the progression in terms of point, green apples and red apples. The success rate is also illustrated using a line chart. With this mode it is possible to clearly see the progression of the patient and how he/she is responding to the therapy.

6.3.3 Configuration (setup Wizard)

Configuration is one of the main objectives of our work. Identified by therapists has one of the biggest gaps in adapted video games, configuration deserves special attention. Therapist report that the video games used nowadays in the physiotherapy process, sin for their inability to configure parameters of the game and for the capacity to adapt to the patient's condition. The configuration has to be supported by the mobile interface and therapist should be able to configure a session using the mobile application.

Since we have two levels with some different configuration parameters, implement two different wizards was not an option, so we needed a setup wizard that can handle both configurations. Figure 32 illustrates the wizard sequence diagram where it is possible to visualise the path taken by the wizard when configuring each one of the levels.

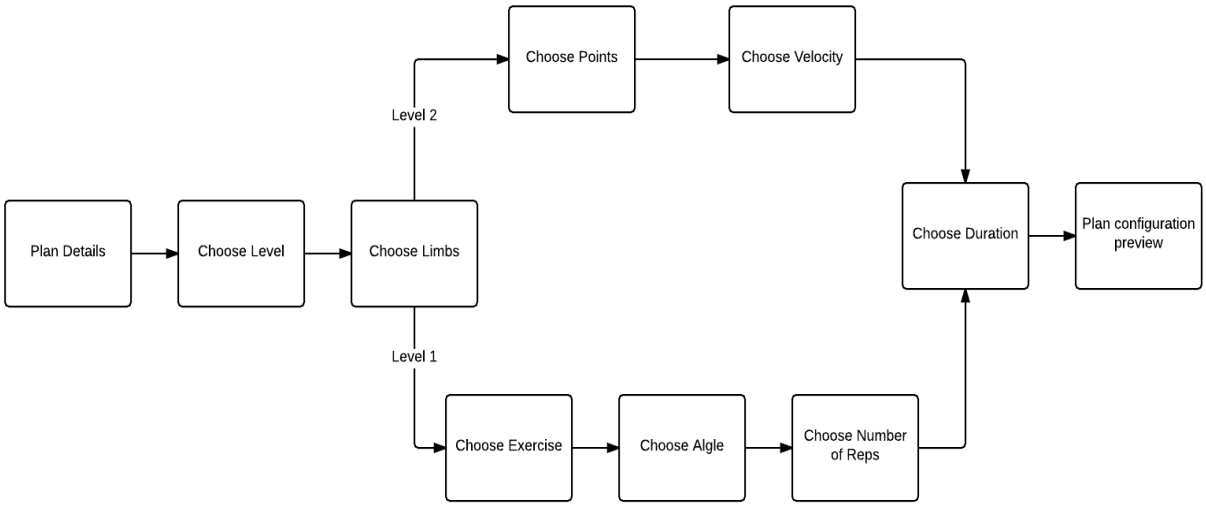


Figure 32 - Configuration Wizard sequence diagram

Regarding to the configuration interface, it should be simple to use and has to speed up the configuration process. For that reason we decide to create a setup wizard, which splits the configuration parameters in different views creating a more friendly and clean GUI. Using the wizard, the therapist has a guide line for the configuration, each one of the parameters is presented one at a time and the values allowed for each one of the parameters are already predefined, the therapist just has to choose the one that fits best for the patient involved. This way the therapist doesn't have to type anything, avoiding typing errors that can cause system errors or incompatibilities. This organization and implementation of the game parameters configuration allows us to speed up this process and contribute to a more efficient and usable application.

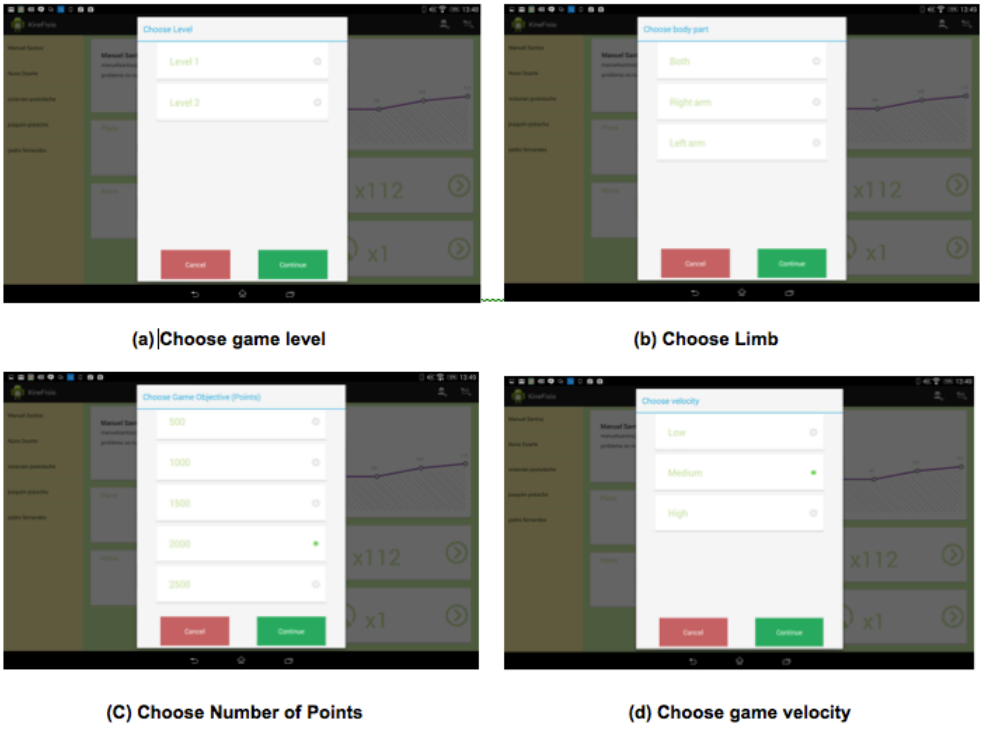


Figure 33 - Game Configuration using setup wizard

Figure 33 illustrates part of the sequence of the setup wizard when configuring a session choosing the game level 2. After select the level 2 the therapist should choose the limb we wants the patient to work with, then choose the game objective selecting the amount of points that the patient must perform to complete a session successfully and then choose the game velocity. As reported earlier, the configuration parameters values are presented by default to prevent typing errors and speed up the process. A visual feedback consisting in a green dot was added to the list elements to facilitate the comprehension of which item is selected.

6.4 Conclusion

In this chapter the structure of the Android OS mobile application is explained. All the UIs are explained, from the top level to the more detailed ones as well as the reasons behind their structure. The approach of the game configuration through a setup wizard is discussed and advantages are presented. The usefulness of a network communication using the Volley library, which allows asynchronous HTTP requests and deals with cache is also explain.

7 Results

In this section, are presented the results associated with the tests made using the Kinect based system and the developed rehabilitation game for different users. The capacity of the designed and implemented system to collect data related to range, as it seen in section 7.1, and velocity, as seen in section 7.2, while patients played the game was tested. The represented data was normalized. The developed mobile interface is also presented with some results concerning the EHR and physiotherapy training data visualization.

We carried out this study with two male participants. Each participant played the pick up apples game during two minutes. We choose a sample from the collected data to evaluate the performance of the system regarding to patient's hands range and velocity. To perform the test, each participant was standing up three meters away in front of the Kinect sensor.

7.1 Range Results

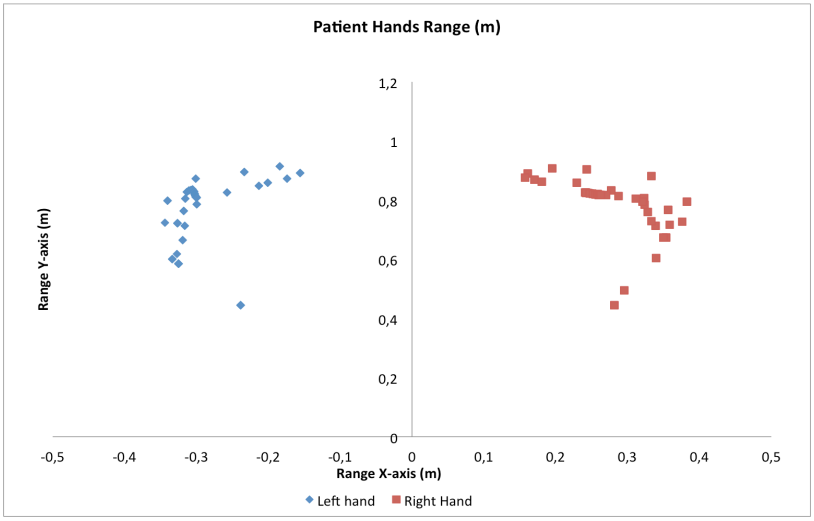


Figure 34 - Left Hand/Right Hand range test patient 1

Figure 34 illustrates the ranges associated to the left and right hands os patient 1. The plot was made using a sampling window chosen from the three-dimensional vectors collected during a session performed by this patient. As seen in the plot it is clear that both hands has a similar behaviour and the range of each hand is pretty much the same, which could mean that the patient 1 presents no problem in terms of mobility of the upper limbs. This happens because patient 1, in fact, has no problems at all, he will be the control group.

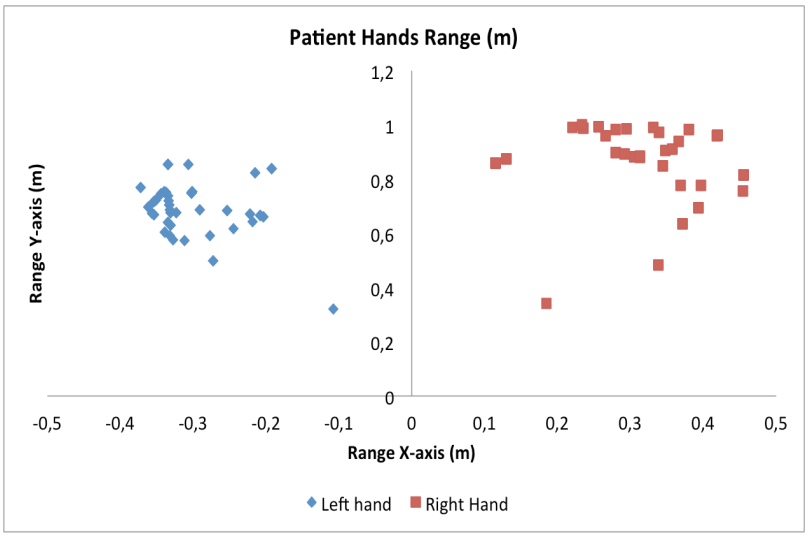


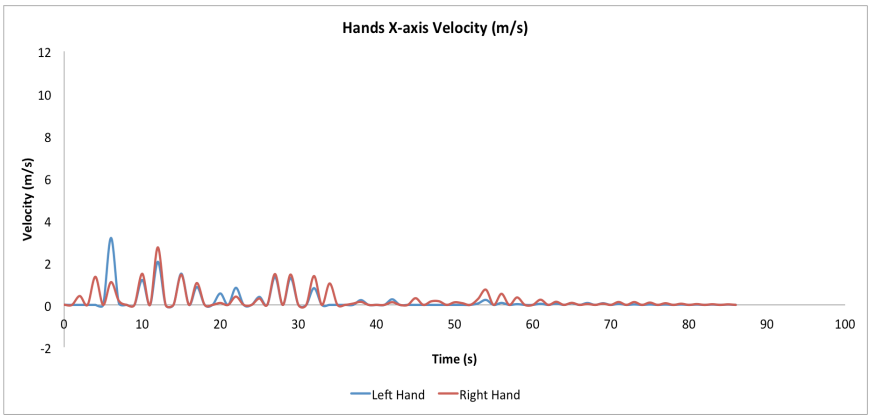
Figure 35 - Left Hand/Right Hand range test patient 2

Similarly, **Figure 35** illustrates the ranges associated to the left and right hands of patient 2. Analysing the plot it is clear that patient 2 hand's ranges are quite different. The patent's right hand has a similar behave when comparing to patient 1 hand's behaviour but the patient 2 left hand presents different pattern of movement, which could indicate that patient 2 left hand presents some mobility problems.

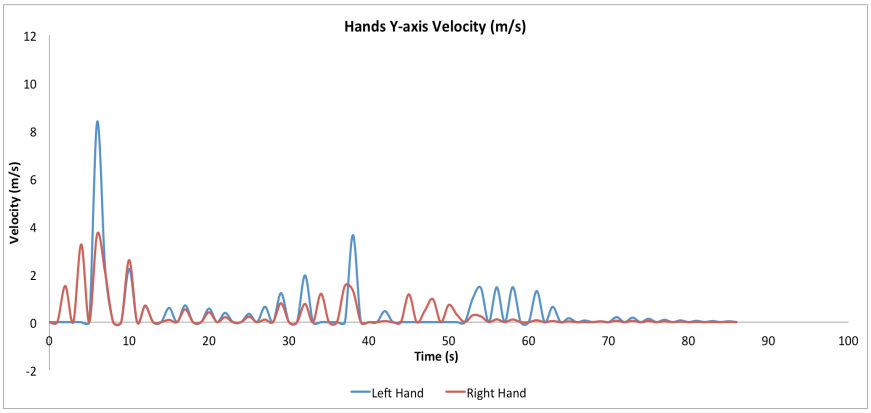
Both figures illustrating the ranges regarding to left hand and right hand motion performed during a rehabilitation session show us that Microsoft Kinect sensor has the capacity to collect sufficient data that can be used to track the evolution of the patient's hands range. Acquiring data from consecutive sessions performed by the same patient may help to analyze the values of the measured ranges during different physiotherapy sessions and to map the patient's progress.

7.2 Velocity Results

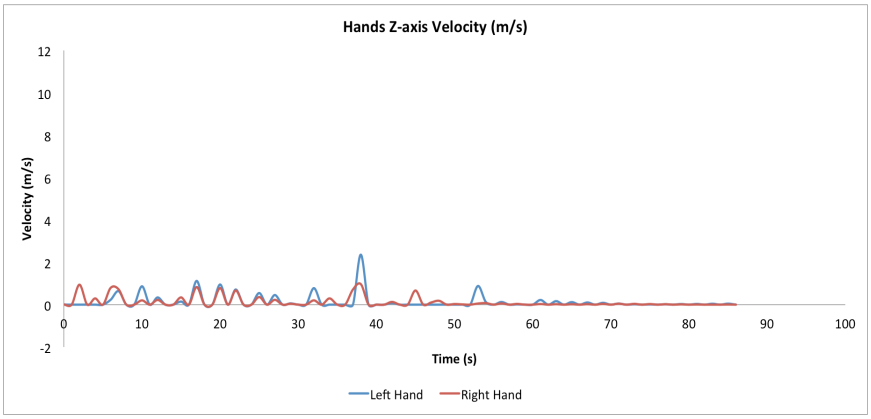
In a similar manner, using the Microsoft Kinect sensor we can calculate the velocity of the left and right hands regarding to X-axis, Y-axis and Z-axis. To ease the comprehension of the results we associate the axis to three movements. A variation over the X-axis represents a lateral extension of the arm, a variation over Y-axis when the patient moves the arm up and down and a variation over Z-axis when the patient moves the arm forwards and backwards. Using the same sampling window used in the range results we calculate and analyse the velocities of both hands for each one of the patients. Given the velocity values in different axis may help to understand the mobility of a patient's arm in different orientations.



(a) Left hand and right hand X-axis velocity



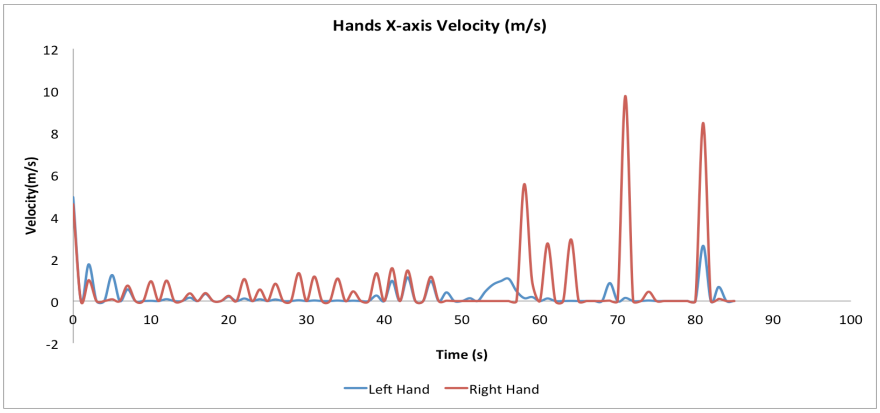
(b) Left hand and right hand Y-axis velocity



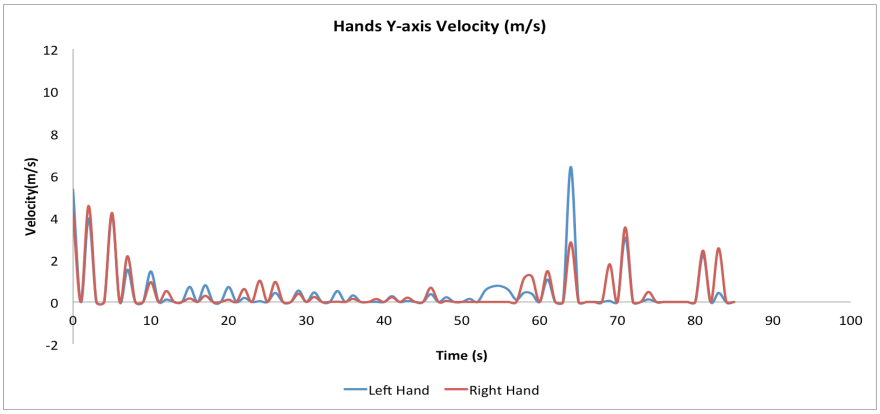
(c) Left hand and right hand Z-axis velocity

Figure 36 - X-axis, Y-axis, Z-axis patient 1 hands velocity performing a game session

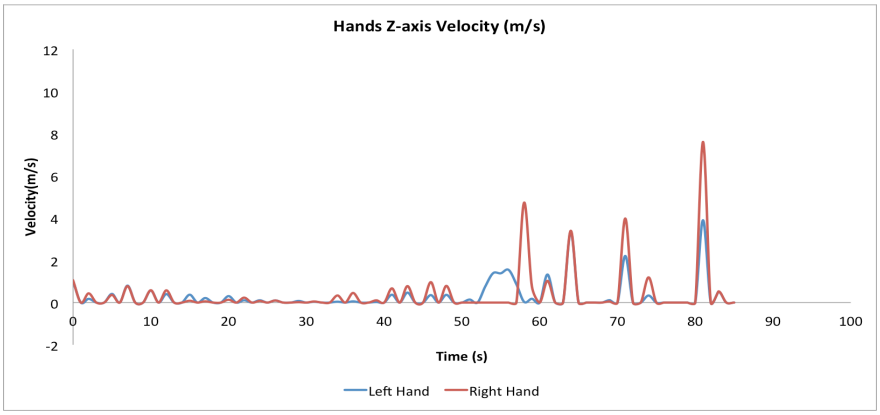
Figure 36 illustrates the patient 1 X-axis, Y-axis and Z-axis velocities respectively. Analysing the plots it is possible to understand how patient 1 moves his arms. Looking to the three plots it is visible that the variation of the velocity over the Y-axis, plot (b), is more pronounced, which indicates that this patient moves his arm up and down quite often than the other movements.



(a) Left hand and right hand X-axis velocity



(b) Left hand and right hand Y-axis velocity



(c) Left hand and right hand Z-axis velocity

Figure 37 - X-axis, Y-axis, Z-axis patient 2 hands velocity performing a game session

Similarly Figure 37 illustrates the patient 2 X-axis, Y-axis and Z-axis velocities respectively. Analysing the three plots it is possible to see that patient 2 unlike patient 1 executes more often an extension of the arms, as seen in Figure 37(a) and that he execute all the three movements simultaneously at a certain point.

7.3 Mobile application statistics

Using the mobile application the therapist has the access to statistics regarding patient's rehabilitation sessions. The figures above illustrate some of that statistics presented by the mobile application using the comparison mode and the single session analysis mode.

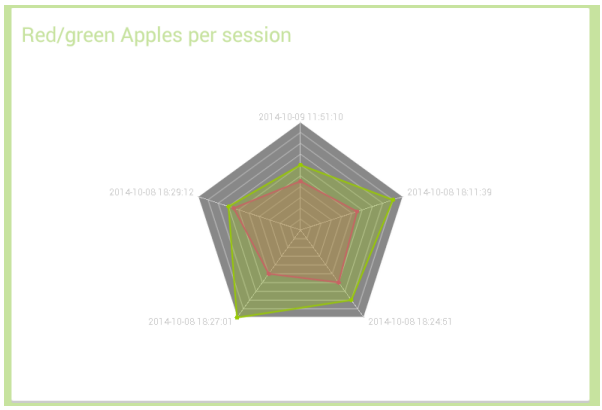


Figure 38 – Red/Green Apples per session comparison

In Figure 38 we can see the comparison between the numbers of objects picked up by the patient in the last five sessions sorted by color. This way the therapist can understand the behaviour of the patient and look for progression signs.

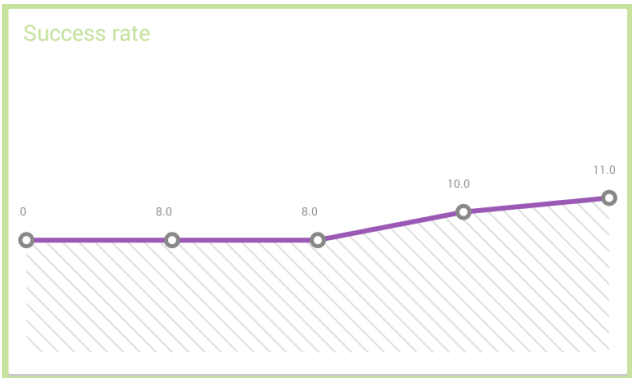


Figure 39 – Success rate comparison

Figure 39 illustrates the comparison between the success rates of the last five sessions, which is a ratio between the number of apples that can be picked up by the patient and the number of objects that he actually picked during the session. This value gives the therapist a global idea of the patient´s performance during a session and about the patient's progression.

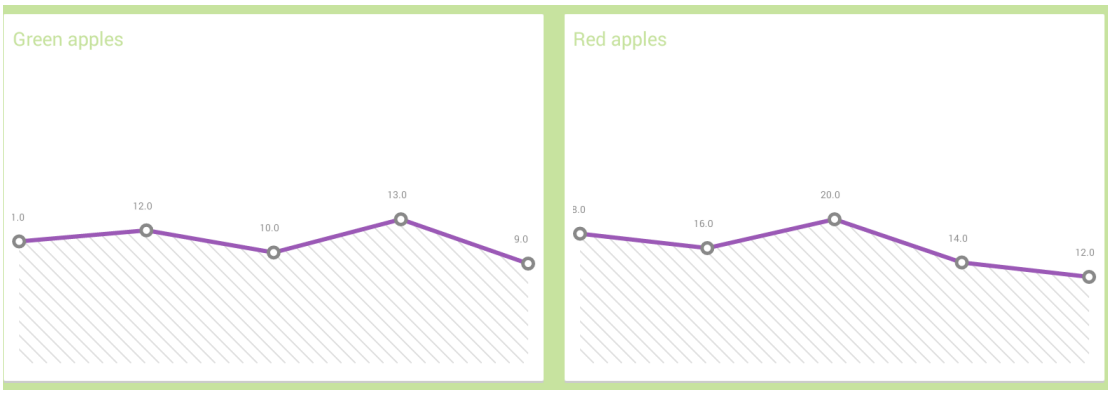


Figure 40 – Comparison between the number of red and green apples picked up in last five sessions

In Figure 40 we can see the user interface showing an overview of the patient's progression in the last five sessions, in terms of the number of red and green apples caught. These charts allow the therapist to understand the evolution of the cognitive capacities of the patient, given that red apples are harder to catch than the green apples.

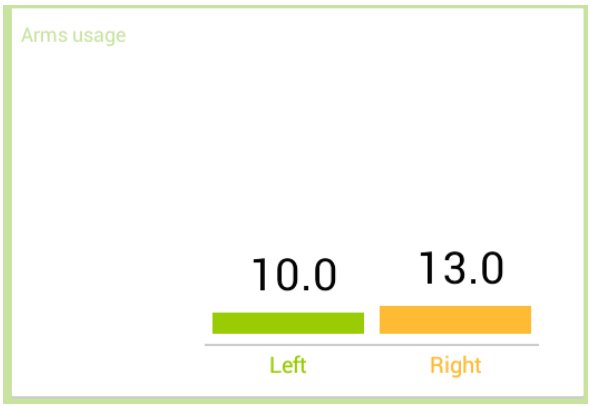


Figure 41 – Arms usage

Figure 41 illustrates the arms usage comparison. This chart belongs so the single session analysis interface and aims to show which arm the patient uses more to catch apples. This way the therapist can understand id the patient's physical and cognitive behaviours.

8 Conclusion and Future Work

8.1 Conclusion

Microsoft Kinect sensor is an appropriate tool for data collection because it enables an appropriate analysis of patient's behaviour during a rehabilitation session. The advantages of combining the Kinect sensor with the rendering engine Unity3D and mobile applications to create 3D serious games capable of being configurable and adapted to the patient's needs are undeniable. Using Unity3D allows us to replicate real life situations, which facilitates the transition between virtual and real world. The mobile application eases the access to the data and the analysis process allowing the therapist to see relevant statistics of a patient session, enables a single therapist to monitor several patients at the same time and allows him to configure specific parameters as velocity or angles of a session according to the patient's needs. Both, game and mobile application have a good acceptance in the therapist community, the idea of something that can store the patient's exercise performances and analyse the patient's behaviour sparked the interest of the majority of therapists. However, many of the therapists manifest their concern about the quality of the movements performed by the patient during the sessions, arguing that the game is rewarding the patient with points every time he catches an apple instead of rewarding the patient every time he/she performs the desired movement correctly.

We performed a test in a therapy clinic to understand how the patient's react to the game in a clinical environment and it seems to be quite adherence mostly to the younger ones. Every time a patient played the game it was possible to see the joy and the enthusiasm and they seem fairly committed to the game but only with one test is difficult to tell if the commitment is real or if it's just because they are experiencing something new.

8.2 Future work

As future work, we pretend to focus on the therapist analysis of the game and work on a solution to solve the quality of the movement issue. The main idea is to develop a module that works as a filter. It receives the three-dimensional vectors that represent the movement performed by the patient and outputs the accuracy of the movement by comparing it to a correct one. This solution must be compatible with Unity3D game engine.

9 References

[1] Michelle Annett, Walter F. Bischo Fraser Anderson, "Lean on Wii: Physical Rehabilitation With Virtual Reality and Wii Peripherals," *Stud Health Technol Inform*, pp. 154:229-34, 2010.

[2] Jun-Da Huang, "Kinerehab: A Kinect-based System for Physical Rehabilitation — A Pilot Study for Young Adults with Motor Disabilities," *Research in Developmental Disabilities Volume 32, Issue 6*, pp. 2566–2570, 11-12 2011.

[3] Bassett.H.S, "The Assessment of Patient Adherence to Physiotherapy Rehabilitation," *New Zealand Journal of Physiotherapy*, pp. 60-66, 2003.

[4] E. Ammenwerth A. Hoerbst, "Electronic Health Records A Systematic Review on Quality Requirements," *Methods Inf Med*, 2010.

[5] A. Burstin, A. Weisel-Eichler, R Brown H. Sugarman, "Use of the Wii Fit System for the treatment of balance problems in the elderly: A case report," *Virtual Rehabilitation*, pp. 111-116, 2009.

[6] D. Robbins, J. Morrison, and P. Guarrera-Bowlby J. Deutsch, "Wii-based compared to standard of care balance and mobility rehabilitation for two individuals post-stroke," *Virtual Rehabilitation*, pp. 117- 120, 2009.

[8] M.D.J. McNeill, D.K. Charles, P.J. Morrow, J.H. Crosbie, S.M. McDonough J.w. Burke, "Optimising engagement for stroke rehabilitation using serious games," *The Visual Computer*, pp. 1085-1099, Aug. 2001.

[7] Josef Wiemeyer. Annika Kliem, "Serious games in prevention and rehabilitation—a new panacea for elderly people? ," *Eur Rev Aging Phys Act* , pp. 9:41–50 , 2012.

[9] Bechkoum, K. Minhua Ma, "Serious games for movement therapy after stroke," *Systems, Man and Cybernetics*, pp. 1872 - 1877, 2008.

[10] Hyung Park, Paolo Bonato, Leighton Chan, Mary Rodgers Shyamal Patel, "A review of wearable sensors and systems with application in rehabilitation," *Journal of NeuroEngineering and Rehabilitation*, pp. 9:21, 2012.

[11] Marion Souila, Abdelmadjid Bouabdallaha, Yacine Challala, Henry Owenb Abdelkrim Hadjidja, "Wireless Sensor Networks for Rehabilitation Applications: Challenges and Opportunities Version 1," *Journal of Network and Computer Applications 36*, pp. 1-15, Dec 2013.

[12] Robert Teasell, Muhammad Mamdani, Judith Hall, William McIlroy, Donna Cheung, Kevin E. Thorpe, Leonardo G. Cohen, Mark Bayley Gustavo Saposnik, "Effectiveness of Virtual Reality Using Wii Gaming Technology in Stroke Rehabilitation: A Pilot Randomized Clinical Trial and Proof of Principle," *American Heart Association*, pp. 41:1477-1484, May 2010.

[13] Lange B. , Mi Zhang , Koenig S. , Requejo P. , Noom Somboon , Sawchuk A.A. , Rizzo A.A. Chien-Yen Chang, "Towards Pervasive Physical Rehabilitation Using Microsoft Kinect," *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2012 6th International Conference*, pp. 159 – 162, 2012.

[14] Eijiro Adachi, Takashi Masuda , Jun-ichiMizusawa NaofumiKitsunezaki, "KINECT Applications for The Physical Rehabilitation," *Medical Measurements and Applications Proceedings (MeMeA), 2013 IEEE International Symposium*, pp. 294 – 299, may 2014.

[15] HGil-Gómez , JAGil-Gómez, SAlbiol-Pérez,GPalacios,HabibM.Fardoum, Abdulfattah S. Mashat, JALozano-Quilis, "Virtual Reality System for Multiple Sclerosis Rehabilitation using KINECT," *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference* , pp. 366 – 369, May 2013.

[16] Jason Blood, Barry Smyth Brian Caulfield, "Rehabilitation exercise feedback on Android platform," *Proceedings of the 2nd Conference on Wireless Health* , 2011.

[18] The story behind Microsoft's hot selling kinect.. [Online]. http://www.businessinsider.com/the-story-behind-microsofts-hot-selling-kinect-2011-1

[17] Dario Maggiorini, Claudio E. Palazzi Dario Deponti, "DroidGlove: An Android-Based Application for Wrist Rehabilitation," *ICUMT '09 International Conference* , pp. 1- 7, Oct 2009.

[19] Abhijit Jana, *Kinect for Windows SDK Programming Guide*., 2012.

[20] Unity3D. [Online]. http://unity3d.com/

[21] RF Solutions. Unity asset store. [Online]. https://www.assetstore.unity3d.com/en/#!/content/7747

[22] theukwebdesigncompany. theukwebdesigncompany. [Online]. http://www.theukwebdesigncompany.com/articles/types-of-databases.php

[23] Oracle. MySQL. [Online]. http://www.mysql.com/

[24] Oracle. MySQL. [Online]. http://www.mysql.com/products/workbench/

[25] David Orenstein. Computerworld. [Online]. http://www.computerworld.com/article/2593623/app-development/application-programming-interface.html

[26] ECMA. Introducing JSON. [Online]. http://json.org/

[28] Google. Android Developers. [Online]. http://developer.android.com/training/volley/index.html

[27] The PHP Group. PHP. [Online]. http://php.net/

[29] Google. Android Developers. [Online]. https://developer.android.com/design/patterns/index.html

Appendices

Appendix A – JointsHandler class

This class is the class responsible for handling the joints. It is used to calculate angles and to communicate with the server.

```
using UnityEngine;
using System;
using System.Collections;
using SimpleJSON;
public class JointsHandler : MonoBehaviour{
        public enum NuiSkeletonPositionIndex : int
        {
                HipCenter = 0,
                Spine,
                ShoulderCenter,
                Head,
                ShoulderLeft,
                ElbowLeft,
                WristLeft,
                HandLeft,
                ShoulderRight,
                ElbowRight,
                WristRight,
                HandRight,
                HipLeft,
                KneeLeft,
                AnkleLeft,
                FootLeft,
                HipRight,
                KneeRight,
                AnkleRight,
```

```
        FootRight,

        Count

}

// Kinect-given Variables to keep track of the skeleton's joints.

public enum SkeletonJoint

{

        //NONE = 0,

        HEAD = NuiSkeletonPositionIndex.Head,

        NECK = NuiSkeletonPositionIndex.ShoulderCenter,

        SPINE = NuiSkeletonPositionIndex.Spine,  // TORSO_CENTER

        HIPS = NuiSkeletonPositionIndex.HipCenter,  // WAIST

        LEFT_COLLAR = -1,

        LEFT_SHOULDER = NuiSkeletonPositionIndex.ShoulderLeft,

        LEFT_ELBOW = NuiSkeletonPositionIndex.ElbowLeft,

        LEFT_WRIST = NuiSkeletonPositionIndex.WristLeft,

        LEFT_HAND = NuiSkeletonPositionIndex.HandLeft,

        LEFT_FINGERTIP = -1,

        RIGHT_COLLAR = -1,

        RIGHT_SHOULDER = NuiSkeletonPositionIndex.ShoulderRight,

        RIGHT_ELBOW = NuiSkeletonPositionIndex.ElbowRight,

        RIGHT_WRIST = NuiSkeletonPositionIndex.WristRight,

        RIGHT_HAND = NuiSkeletonPositionIndex.HandRight,

        RIGHT_FINGERTIP = -1,

        LEFT_HIP = NuiSkeletonPositionIndex.HipLeft,

        LEFT_KNEE = NuiSkeletonPositionIndex.KneeLeft,

        LEFT_ANKLE = NuiSkeletonPositionIndex.AnkleLeft,

        LEFT_FOOT = NuiSkeletonPositionIndex.FootLeft,

        RIGHT_HIP = NuiSkeletonPositionIndex.HipRight,

        RIGHT_KNEE = NuiSkeletonPositionIndex.KneeRight,

        RIGHT_ANKLE = NuiSkeletonPositionIndex.AnkleRight,

        RIGHT_FOOT = NuiSkeletonPositionIndex.FootRight,


        END
```

```
        };
        //
        private const int head = (int)SkeletonJoint.HEAD;
        private const int neck = (int)SkeletonJoint.NECK;
        private const int spine = (int)SkeletonJoint.SPINE;
        private const int leftHandIndex = (int)SkeletonJoint.LEFT_HAND;
        private const int rightHandIndex = (int)SkeletonJoint.RIGHT_HAND;
        private const int leftElbowIndex = (int)SkeletonJoint.LEFT_ELBOW;
        private const int rightElbowIndex = (int)SkeletonJoint.RIGHT_ELBOW;
        private const int leftShoulderIndex = (int)SkeletonJoint.LEFT_SHOULDER;
        private const int rightShoulderIndex = (int)SkeletonJoint.RIGHT_SHOULDER;
        private const int hipCenterIndex = (int)SkeletonJoint.HIPS;
        private const int shoulderCenterIndex = (int)SkeletonJoint.NECK;
        private const int leftHipIndex = (int)SkeletonJoint.LEFT_HIP;
        private const int rightHipIndex = (int)SkeletonJoint.RIGHT_HIP;
        Vector3[] jointsPos ;
        private AngleCalculator ac ;
        string url = "http://193.136.221.214/kinect_scripts/kinect.php?action=newscene&id=";
        string addUrl  =
"http://193.136.221.214/kinect_scripts/kinect.php?action=insertdata&id=";
        string sceneID="";
        string sessionID="";

        // angles left elbow
        int leftElbowAngle = 0;
        bool calcLeftElbowAngle = false;
        Vector3 leftElbowPos;

        // angles left shoulder
        int leftShoulderAngle = 0;
        bool calcLeftShoulderAngle = false;
        Vector3 leftShoulderPos;
```

```
        // angles right elbow
        int rightElbowAngle = 0;
        bool calcRightElbowAngle = false;
        Vector3 rightElbowPos;

        // angles right shoulder
        int rightShoulderAngle = 0;
        bool calcRightShoulderAngle = false;
        Vector3 rightShoulderPos;
        GUIStyle largeFont;
        int config_angle ;
        bool save = false;  // used to define if the game should or not store data
        bool armUp = false;
        bool armDown = true;

        void Start(){
                ac = new AngleCalculator (); // instantiation of the angle calculator object
                sessionID = PlayerPrefs.GetString ("sessionid");
                largeFont = new GUIStyle();
                largeFont.fontSize = 40;
                largeFont.normal.textColor = Color.white;
                config_angle=PlayerPrefs.GetInt ("config_angle");
        }
        void OnGUI(){
                //if (calcLeftElbowAngle)
                //      GUI.Label (new Rect (Screen.width - 500,40,200,200),"Left Elbow
"+leftElbowAngle+" °" , largeFont); ----------- LEVEL 2
                //if (calcLeftShoulderAngle) ----------------------- LEVEL 2
                //      GUI.Label (new Rect (Screen.width - 500,80,200,200), "Left Shoulder
"+leftShoulderAngle+" °" , largeFont); ---------- LEVEL 2
        }
        public void setJoinPos( ref Vector3[]jPos){
                //Debug.Log ("setjonts -- "+jPos.ToString());
```

```
                this.jointsPos = jPos;
                if(sessionID != "" && save){
                        createScene();
                }
        }
        public void getRigthElbowAngle(){
                Vector3 handPos = jointsPos[rightHandIndex];
                rightElbowPos = jointsPos [rightElbowIndex];
                Vector3 showPos = jointsPos[rightShoulderIndex];
                if(ac!=null)
                        rightElbowAngle = (int)ac.getAngle (handPos, rightElbowPos, showPos);
                //Debug.Log ("Angle left elbow: " + leftElbowAngle);
                if(rightElbowAngle >= config_angle-5 && rightElbowAngle <=config_angle+5 &&
armDown){
                        armUp=true;
                        armDown = false;
                }
                if (rightElbowAngle >= 165 && armUp) {
                        Debug.Log ("flexao do cotovelo direito!");
                        GameObject.Find("scripts").SendMessage ("addRep");
                        armUp = false;
                        armDown = true;
                }
        }
        public void getLeftElbowAngle(){
                if(!calcLeftElbowAngle){
                        calcLeftElbowAngle = true;
                }
                Vector3 handPos = jointsPos[leftHandIndex];
                leftElbowPos = jointsPos [leftElbowIndex];
                Vector3 showPos = jointsPos[leftShoulderIndex];

                if(ac!=null)
```

```
                leftElbowAngle = (int)ac.getAngle (handPos, leftElbowPos, showPos);
                Debug.Log ("Angle left elbow ---> " + leftElbowAngle);
                if(leftElbowAngle >= config_angle-5 && leftElbowAngle <=config_angle+5 &&
armDown){
                        armUp=true;
                        armDown = false;
                }
                if (leftElbowAngle >= 165 && armUp) {
                        Debug.Log ("flexao do cotovelo esquerdo!");
                        GameObject.Find("scripts").SendMessage ("addRep");
                        armUp = false;
                        armDown = true;
                }
        }
        public void getRightShoulderAngle(){
                if(!calcRightShoulderAngle){
                        calcRightShoulderAngle = true;
                }
                Vector3 handPos = jointsPos[rightHandIndex];
                Vector3 hipPos = jointsPos [rightHipIndex];
                rightShoulderPos = jointsPos[rightShoulderIndex];
                if(ac!=null)
                        rightShoulderAngle = (int)ac.getAngle (handPos, rightShoulderPos,
hipPos);
                Debug.Log ("Right shoulder angle ---> "+rightShoulderAngle);
                if(rightShoulderAngle >= config_angle-5 && rightShoulderAngle <=config_angle+5
&& armDown){
                        armUp=true;
                        armDown = false;
                }
                if (rightShoulderAngle <= 30 && armUp) {
                        Debug.Log ("abduçao do obmbro direito!");
                        GameObject.Find("scripts").SendMessage ("addRep");
                        armUp = false;
```

```
                armDown = true;
            }
        }
        public void getLeftShoulderAngle(){
            if(!calcLeftShoulderAngle){
                calcLeftShoulderAngle = true;
            }
            Vector3 handPos = jointsPos[leftHandIndex];
            Vector3 hipPos = jointsPos [leftHipIndex];
            leftShoulderPos = jointsPos[leftShoulderIndex];
            if(ac!=null)
                leftShoulderAngle = (int)ac.getAngle (handPos, leftShoulderPos,
hipPos);
            if(leftShoulderAngle >= config_angle-5 && leftShoulderAngle <=config_angle+5 &&
armDown){
                armUp=true;
                armDown = false;
            }
            if (leftShoulderAngle <= 30 && armUp) {
                Debug.Log ("abduçao do obmbro esquerdo!");
                GameObject.Find("scripts").SendMessage ("addRep");
                armUp = false;
                armDown = true;
            }
        }
        public void createScene(){
            StartCoroutine (getSceneID());
        }
        public void addJointsToScene(string ID){
            addHead (ID);
            addNeck (ID);
            addSpine (ID);
            addLeftShoulder (ID);
```

```
        adRightShoulder (ID);
        addLeftElbow (ID);
        addRightElbow (ID);
        addLeftHand (ID);
        addRightHand (ID);
}
public void addHead(string id){
        Vector3 headPos = jointsPos[head];
        String headUrl = addUrl + id + "&bpid=5";
        Debug.Log (headUrl);
        WWWForm form = new WWWForm();
        form.AddField("x", headPos.x.ToString());
        form.AddField("y", headPos.y.ToString());
        form.AddField("z", headPos.z.ToString());
        WWW www = new WWW(headUrl, form);
        //Debug.Log (www.ToString());
        StartCoroutine (addJointsToScene (www));
}
public void addNeck(string id){
        Vector3 neckPos = jointsPos[neck];
        String neckUrl = addUrl + id + "&bpid=4";
        WWWForm form = new WWWForm();
        form.AddField("x", neckPos.x.ToString());
        form.AddField("y", neckPos.y.ToString());
        form.AddField("z", neckPos.z.ToString());
        WWW www = new WWW(neckUrl, form);
        StartCoroutine (addJointsToScene (www));
}
public void addSpine(string id){
        Vector3 spinePos = jointsPos[spine];
        String spineUrl = addUrl + id + "&bpid=3";

        WWWForm form = new WWWForm();
```

```
        form.AddField("x", spinePos.x.ToString());
        form.AddField("y", spinePos.y.ToString());
        form.AddField("z", spinePos.z.ToString());
        WWW www = new WWW(spineUrl, form);
        StartCoroutine (addJointsToScene (www));
    }

    public void addLeftShoulder(string id){
        Vector3 showPos = jointsPos[leftShoulderIndex];
        String showUrl = addUrl + id + "&bpid=7";
        WWWForm form = new WWWForm();
        form.AddField("x", showPos.x.ToString());
        form.AddField("y", showPos.y.ToString());
        form.AddField("z", showPos.z.ToString());
        WWW www = new WWW(showUrl, form);
        StartCoroutine (addJointsToScene (www));
    }
    public void addRightShoulder(string id){
        Vector3 showPos = jointsPos[rightShoulderIndex];
        String showUrl = addUrl + id + "&bpid=12";
        WWWForm form = new WWWForm();
        form.AddField("x", showPos.x.ToString());
        form.AddField("y", showPos.y.ToString());
        form.AddField("z", showPos.z.ToString());
        WWW www = new WWW(showUrl, form);
        StartCoroutine (addJointsToScene (www));
    }

    public void addLeftElbow(string id){
        Vector3 elbowPos = jointsPos[leftElbowIndex];
        String elbowUrl = addUrl + id + "&bpid=8";
        WWWForm form = new WWWForm();
        form.AddField("x", elbowPos.x.ToString());
```

```
        form.AddField("y", elbowPos.y.ToString());
        form.AddField("z", elbowPos.z.ToString());
        WWW www = new WWW(elbowUrl, form);
        StartCoroutine (addJointsToScene (www));
}
public void addRightElbow(string id){
        Vector3 elbowPos = jointsPos[rightElbowIndex];
        String elbowUrl = addUrl +id+ "&bpid=13";
        WWWForm form = new WWWForm();
        form.AddField("x", elbowPos.x.ToString());
        form.AddField("y", elbowPos.y.ToString());
        form.AddField("z", elbowPos.z.ToString());
        WWW www = new WWW(elbowUrl, form);
        StartCoroutine (addJointsToScene (www));
}
public void addLeftHand(string id){
        Vector3 handPos = jointsPos[leftHandIndex];
        String handUrl = addUrl + id + "&bpid=10";
        WWWForm form = new WWWForm();
        form.AddField("x", handPos.x.ToString());
        form.AddField("y", handPos.y.ToString());
        form.AddField("z", handPos.z.ToString());
        WWW www = new WWW(handUrl, form);
        StartCoroutine (addJointsToScene (www));
}

public void addRightHand(string id){
        Vector3 handPos = jointsPos[rightHandIndex];
        String handUrl = addUrl + id + "&bpid=15";
        WWWForm form = new WWWForm();
        form.AddField("x", handPos.x.ToString());
        form.AddField("y", handPos.y.ToString());
        form.AddField("z", handPos.z.ToString());
```

```
        WWW www = new WWW(handUrl, form);
        StartCoroutine (addJointsToScene (www));
    }
    // create a new scene
    IEnumerator getSceneID(){
        string newscene = url + sessionID;
        //Debug.Log (url);
        WWW request = new WWW(newscene);
        yield return request;
        if (request.error == null || request.error == "")
        {
            var N = JSON.Parse(request.text);
            sceneID = N["scene"][0]["id"];
            Debug.Log("ID do ultimo scene : " + sceneID);
            addJointsToScene(sceneID);
        }
        else
        {
            Debug.Log("WWW error: " + request.error);
        }
    }
    // add a joit to scene
    IEnumerator addJointsToScene(WWW request){
        yield return request;
        if (request.error == null || request.error == "")
        {
            Debug.Log("WWW ok");
        }
        else
        {
            Debug.Log("WWW error: " + request.error);
        }}
```

Appendix B – API documentation

API Structure:

- Kinect.php (root script)

- User folder
  - User.php
- History
  - History.php
- Login
  - Login.php
- Notes
  - Notes.php
- Plans
  - Plans.php
- Sessions
  - Sessions.php

Language used – PHP

Access to database – PDO (PHP Data Objects) Link: http://php.net/manual/en/book.pdo.php

User folder

The user folder contains the User.php script that aggregates all the methods needed to handle the users data. A description of each method is presented as well as an example of the requests and responses.

addUser --

This method is used to add new users to the database. To successful add a new user we need a set of arguments that must be send with the request.

Arguments:

$name – The user first name.

$email – The user email.

$surname – The user surname.

$address – The user address.

$birth_date – The user birthday date.

$gender - The user gender male or female is replaced by an integer when stored in the database [male = 1 , female = 0].

$description – A description of the user problem if he is a patient.

$type - The type of user, defines if the user is a patient or a therapist represented by an integer [patient = 0 , physiotherapist= 1].

$parent_user – This arguments set a relation between therapists and patients, this way we can associate multiple patients to a single therapist.

Request:

WEBROOT_PATH/kinect.php?action=newuser&mail=<email>&name=<firstname>&sname=<lastname>&addr=<useraddress>&bday=<birthdaydate>&gen=<gender>&type=<type>&pid=<parentID>

Response:

If something went wrong the system will return a "0".

listPatients ----------

This method is used to return the therapist's patient list.

Arguments:

$parent_ID – The ID of the therapist which we want the patient list.

Request:

WEBROOT_PATH/kinect.php?action=listpatients&pid=<parentID>

Response:

{"users":[
{
"name":"Manuel",
"surname":"Santos",
"id":"7"
},
{
"name":"Nuno",
"surname":"Duarte",
"id":"8"
}
]}

```
userInfo ------------------------------------------------

This method returns the data related to a specific user.

Arguments:

$userID - The ID of the user which we want the data

Request:

WEBROOT_PATH/kinect.php?action=userinfo&id=<userID>


Response:

{"details":[

        {

        "name":"Manuel",

        "surname":"Santos",

        "email":"manuelsantos@gmail.com",

        "description":"problema no no bra\u00e7o direito",

        "gender":"1"

        }

]}

deleteUser ----------------------------------------------

This method is used to delete the user from the database.

Arguments:

$userID - The ID of the user which we want to delete data

Request:

WEBROOT_PATH/kinect.php?action=deluser&id=<userID>


Response:

Return "0" if something went wrong and return "1" if the used was deleted successfully
```

Notes folder

This folder contains the Notes.php script which is used to handle the notes written by the therapist.

newNote ----------

This method is used to add notes to the database. Every time an application needs to add a note, this method will be used.

Arguments:

$title – The note title.

$content – The note content.

$userID – The patient ID to which the rote referred.

$authorID – The therapist ID who wrote the note.

Request:

Use the POST method.

Response:

Returns "0" if something went wrong.

listNotes ----------

This method is used to list the notes that were written about a patient.

Arguments:

$userID – The ID of the patient.

Request:

WEBROOT_PATH/kinect.php?action=listnotes&id=<UserID>

Response:

```
{"notes":[
        {
        "title":"Problema no antebraço esquerdo",
        "id":"4"
        },
        {
        "title":"Melhorias na última sessão ",
        "id":"5"
        }
]}
```

countNotes ----------

This method returns the number of notes that a patient has associated to his profile.

Arguments:

$userID – The ID of the patient.

Request:

WEBROOT_PATH/kinect.php?action=countnotes&id=<UserID>

Response:

{"notes":[
{
"count":2
}
]}

noteInfo ----------

This method is used to return the details of a specific note.

Arguments:

$noteID – The ID of the note which we want the details.

Request:

WEBROOT_PATH/kinect.php?action=noteinfo&id=<noteID>

Response:

{"note":[
{
"title":"Problema no antebraço esquerdo",
"content":"O paciente apresenta um problema de mobilidade do braço esquerdo",
"date":"2014-05-27 16:16:56",
"author":"Nuno"
}
]}

deleteNote ----------

This method is used to delete a notes related to a patient ID.

Arguments:

$noteID – The ID of the note which we want to delete.

Request:

WEBROOT_PATH/kinect.php?action=delnote&id=<noteID>

Response:

Return "0" if something went wrong and return "1" if the used was deleted successfully

Plans folder

This folder contains the Plans.php script which is used to handle the plans configured by the therapist.

newPlan --

This method is used to add a plan to the database. Every time an application needs to add a plan, this method will be used. A plan is composed by the plan details and the configuration details. After create the plan it will use the plan ID to add a new configuration using the method configure.

Arguments:

$userID – ID of the patient which the plan belongs to.

$title - Plan's title.

$description – Plan's description, the plan main objectives.

Request:

Use the POST method.

Response:

Returns "0" if something went wrong.

listPlans --

This method is used to list the plans that were defined for patient.

Arguments:

$userID – The ID of the patient.

Request:

WEBROOT_PATH/kinect.php?action=listplans&id=<userID>

Response:

```
{"plans":[
        {
        "title":"plano 1",
        "id":"1"
        },
        {
        "title":"Recuperação do braço esquerdo",
        "id":"9"
```

```
        },
        {
        "title":"Continuação da recuperação",
        "id":"19"
        },
        {
        "title":"Teste final",
        "id":"21"
        },
        {
        "title":"test1234",
        "id":"22"
        }
]}
```


countPlans ----------

This method returns the number of plans that a patient has associated to his profile.

Arguments:

$userID - The ID of the patient.

Request:

WEBROOT_PATH/kinect.php?action=countplans&id=<UserID>

Response:


```
{"plans":[
        {
        "count":4
        }
]}
```


planInfo ----------

This method is used to return the details of a specific plan.

Arguments:

$planID - The ID of the plan which we want the details.

Request:

WEBROOT_PATH/kinect.php?action=planinfo&id=<planID>

Response:

```
{"plans":[
        {
        "title":"recuperacao do braco direito",
        "description":"elevacao do ombro a 90 graus",
        "bdate":"2014-07-18 16:07:35",
        "edate":"2014-08-18 16:07:35"
        }
]}
```

getUserLastPlan ----------

This method returns the user last plans ID. It is used in the game application to get the game configuration.

Arguments:

$userID – The ID of the patient.

Request:

WEBROOT_PATH/kinect.php?action=getlastplan&id=<userID>

Response:

```
{"lastPlan":[
        {
        "id":"22"
        }
]}
```

deletePlan ----------

This method is used to delete a plans.

Arguments:

$planID – The ID of the plan which we want to delete.

Request:

WEBROOT_PATH/kinect.php?action=delplan&id=<planID>

Response:

Return “0” if something went wrong and return “1” if the used was deleted successfully

configure ----------

This method is used to add the configuration to a plan.

Arguments:

$planID – The ID of the plan which this configuration belongs.

$big_date – Plan's start date, in timestamp.

$end_date – Plan's end date in timestamp.

$body_partID – ID of the patient body part that he/she should use to execute the exercises.

$min_points – Minimum number of points defined by the therapist to successful end the game.

$angle – The angle that must be performed by the patient during the session.

$velocity - Velocity of the game [slow = 0 , medium = 1 , fast = 2]

$midPointRotation – Dedine if the user should execute the rotation of the body. [true = 1 , false = 0]

$sitting – Define if the patient is sited or not. [true = 1 , false = 0]

Request:

It is called in the newPlan method.

Response:

Return "0" if something went wrong.

getPlanConfiguration ----------

This method is used to get a plan's configuration. Is used by the game application to get the configuration of a session.

Arguments:

$planID – The ID of the plan which we want to get the configuration.

Request:

WEBROOT_PATH/kinect.php?action=getconfiguration&id=<planID>

Response:

{"config":[

{

"angle":"70",

"bodypart":"11",

"points":"1000",

"velocity":"2",

"rotation":"1",

"sitting":"0",

"duration":"60"

}

]}

Login folder

This folder contains the login.php script, which is used to handle the login action.

login ----------

This method is used to login a user in the system. Every time an application needs to add a note, this method will be used.

Arguments:

$username – The user name.

$password – The user password.

Request:

WEBROOT_PATH/kinect.php?action=login&u=<username>&p=<password>

Response:

If login successful :

{"logged":[

{

"id":"4"

}

]}

If error occurs :

{"logged":[

{

"id":"0"

}

]}

History folder

This folder contains the History.php script, which is used to handle the patient medical history.

addClinicalHistory ------------------------------------

This method is used to add the clinical history of a patient.

Arguments:

$user_id – The patient ID.

$disease – The disease name.

$startDate – Defines the start period of the disease in timestamp.

$endDate – Defines if the patient no longer has the disease in timestamp.

$chronic – Defines if the disease is chronic or not. chronic [true = 1, false = 0]

Request:

WEBROOT_PATH/kinect.php?action=newclinicalhistory&uid=<userID>&d=<diseaseName>&sdate=<startDate>&edate=<endDate>&c=<chronic>

Response:

Returns "0" if something went wrong.

getUserClinicalHistory ----------------------------------

This method is used to list the patient clinical history.

Arguments:

$userID – The patient ID.

Request:

WEBROOT_PATH/kinect.php?action= getclinicalhistory&id=<userID>

Response:

{"clinicalHistory":[

{

"ID":"1",

"disease":"artrite",

"startDate":"0000-00-00 00:00:00",

"endDate":"0000-00-00 00:00:00",

"chronic":"1"

},

{

"ID":"2",

"disease":"egegvue",

"startDate":"1990-12-01 00:00:00",

"endDate":"2014-08-30 00:00:00",

"chronic":"0"

}

]}

addMedicationHistory ----------

This method is used to add the medication history of a patient.

Arguments:

$user_id – The patient ID.

$medicine – The medicine name.

$dosage – Defines dosage.

$applicationType – Defines the medicine application type. [oral = 1 , subcutaneous = 2 , injection = 3 , per vagina = 4 , per rectum = 5 , topical = 6]

Request:

WEBROOT_PATH/kinect.php?action=newmedicationhistory&uid=<userID>&m=<medicineName>&d=<dosage>&ap=<aplicationType>

Response:

Returns "0" if something went wrong.

getUserMedicationHistory ----------

This method is used to list the patient medication history.

Arguments:

$userID – The patient ID.

Request:

WEBROOT_PATH/kinect.php?action= getmedicinehistory&id=<userID>

Response:

{"medicationHistory":[

{

"ID":"1",

"medicine":"paracetamol",

```
        "dosage":"500",
        "applicationType":"1"
        },
        {
        "ID":"2",
        "medicine":"Vizacor",
        "dosage":"1000",
        "applicationType":"1"
        },
        {
        "ID":"3",
        "medicine":"clonazepam",
        "dosage":"0.5",
        "applicationType":"1"
        },
        {
        "ID":"4",
        "medicine":"Relaflex",
        "dosage":"50",
        "applicationType":"1"
        },
        {
        "ID":"5",
        "medicine":"ZOMETA",
        "dosage":"4",
        "applicationType":"3"
        }
]}
```


addTherapyHistory ----------

This method is used to add the therapy history of a patient.

Arguments:

$user_id – The patient ID.

$problem – description of the problem.

$startDate – Defines the start period of the problem in timestamp.

$endDate – Defines if the patient no longer has the problem in timestamp.

Request:

WEBROOT_PATH/kinect.php?action=newmtherapyhistory&uid=<userID>&p=<problemName>&sdate=<startDate>&edate=<endDate>

Response:

Returns "0" if something went wrong.

getUserTherapyHistory ------------------------------

This method is used to list the patient therapy history.

Arguments:

$userID – The patient ID.

Request:

WEBROOT_PATH/kinect.php?action= gettherapyhistory&id=<userID>

Response:

{"ThrerapyHistory":[
{
"ID":"1",
"problem":"artrite",
"startDate":"1990-09-12 00:00:00",
"endDate":"0000-00-00 00:00:00"
}
]}

Sessions folder

This folder contains the sessions.php script which is used to handle the all the sessions data.

newSession ----------

This method is used add a new session. Every time the patient starts a game session, the game application executes this request.

Arguments:

$planID – The ID of the plan to which the session belongs.

Request:

WEBROOT_PATH/kinect.php?action=newsession&pid=<plan ID>

Response:

If successful :

```
{"session":[
        {
        "id":"4"
        }
]}
```

If error occurs :

```
{"session":[
        {
        "id":"0"
        }
]}
```

gameSessions ----------

This method is used add a new session. Every time the patient starts a game session, this method is called.

Arguments:

$planID – The ID of the plan to which the session belongs.

Request:

WEBROOT_PATH/kinect.php?action=gamesessions&pid=<plan ID>

Response:

If successful :

```
        {"sessions":[
                {
                "id":"4",
                "date":"2014-08-12"
                }
        ]}
If error occurs :
        Return 0;


sessionData -------------------------------------------
This method returns the data from a specific session
Arguments:
$sessionID – The ID of the session.
Request:
WEBROOT_PATH/kinect.php?action=sessiondata&id=<session ID>
Response:
If successful :
        {"data":[
                {
                "velocity":"2",
                "points":"1100",
                "total_red":"8",
                "total_green":"6",
                "totalApples":"14",
                "progress":"1"
                }
        ]}
If error occurs :


        Returns 0;
```

```
newScene ------------------------------------------

This method is used to add a new scene to the session.

Arguments:

$sessionID – The ID of the session to which the scene belongs.

Request:

WEBROOT_PATH/kinect.php?action=newscene&pid=<session ID>

Response:

If successful :

        {"scene":[

                {

                "id":"3023",

                }

        ]}

If error occurs :

        Returns 0;

insertData ------------------------------------------

This method is used to store data relative to patient's joints for each scene.

Arguments:

$sceneID – The ID of the scene to which data belongs.

$b_partID _ The ID of the body part to which data belongs.

$positionX - The x position of the joint.

$positionY – The y position of the joint.

$positionZ – The z position of the joint.

Request:

WEBROOT_PATH/kinect.php?action=insertdata&id=<scene ID>&bpid=<Body part ID>&x=<X>&y=<Y>&z=<Z>

Response:

If successful :

        Return 1;

If error occurs :

        Returns 0;
```

getLastSessionsData ------------------------------------

This method returns the data from the last five sessions for a specific patient.

Arguments:

$id – The patient's ID;

Request:

WEBROOT_PATH/kinect.php?action=getlastsessionsdata&id=<patient ID>

Response:

If successful :

{"sessions":[

{

"date":"2014-10-06 17:18:33",

"id":"113",

"points":"1000",

"red":"5",

"green":"10",

"total":"128",

"velocity":"2",

"progress":"1"

},{

"date":"2014-10-06 17:05:03",

"id":"112",

"points":"1000",

"red":"6",

"green":"8",

"total":"128",

"velocity":"2",

"progress":"1"

},{

"date":"2014-10-06 17:05:03",

"id":"112",

"points":"1050",

"red":"8",

"green":"5",

```
        "total":"160",

        "velocity":"2",

        "progress":"1"

        },{

        "date":"2014-10-06 17:05:03",

        "id":"112",

        "points":"1000",

        "red":"7",

        "green":"6",

        "total":"160",

        "velocity":"2",

        "progress":"1"

        },{

        "date":"2014-10-06 16:08:37",

        "id":"110",

        "points":"1050",

        "red":"5",

        "green":"11",

        "total":"192",

        "velocity":"2",

        "progress":"1"

        }

]}
```


If error occurs :

        Returns 0;

getUserSessions ----------

This method returns the name of the patient and the session’s data and id.

Arguments:

$ID – The patient ID.

Request:

WEBROOT_PATH/kinect.php?action=getusersessions&id=<patient ID>

Response:

If successful :

```
{"sessions":[{
        "username":"Manuel",
        "date":"2014-05-27 17:51:24",
        "id":"1"
        },{
        "username":"Manuel",
        "date":"2014-05-27 17:51:42",
        "id":"2"
        },{
        "username":"Manuel",
        "date":"2014-05-27 17:51:45",
        "id":"3"
        }
]}
```

If error occurs :

Returns 0;

countSessions ----------

This method returns the number of sessions associated to a patient.

Arguments:

$ID – The patient ID.

Request:

WEBROOT_PATH/kinect.php?action=countsessions&id=<patient ID>

Response:

If successful :

```
        {"sessions":[{
                "count":"20",
        ]}
```

If error occurs :

        Returns 0;

grabAppleEvent ----------

This method is used to store events from the game. When the player catches an object, the game send a grabAppleEvent request to the server.

Arguments:

$type – The type of apple [red = 1 , green = 0].

$timestamp – The timestamp of the event.

$sessionID – The ID of the session to which the event belongs.

$bodypartID – The body part that triggered the event.

Request:

WEBROOT_PATH/kinect.php?action=addappleevent&type=<ObjectType>&time=<EventTimestamp>&sid=<sessionID>&bpid=<BodyPartID>

Response:

If successful :

        Returns 1;

If error occurs :

        Returns 0;

storeProcessedData ----------

This method is used to store events from the game. When the player catches an object, the game send a grabAppleEvent request to the server.

Arguments:

$sessionID – The session Id to which data belongs.

$avelocity – Average movement velocity.

$points – Total of points.

$red – Total of red apples caught.

$green – Total of green apples caught.

$total – Total of apples caught.

$progress – The progress associated to the patient performance.

Request:

WEBROOT_PATH/kinect.php?action=storeprocesseddata&sid=<sessionID>&points=<numberOfPoints>&red=<redApples>&green=<greenApples>&total=<totalApples>&prog=<progression>

Response:

If successful :

Returns 1;

If error occurs :

Returns 0;

getProcessedData ----------

This method returns the processed data for a specific session. Process data represents the final count of the events occurred during a therapy session.

Arguments:

$sessionID – The ID of the session to which the event belongs.

Request:

WEBROOT_PATH/kinect.php?action= getprocesseddata&id=<session ID>

Response:

If successful :

{"data":[
{
"id":"1649",
"avelocity":"2",
"points":"1000",
"red":"5",
"green":"10",
"total":"128",
"prog":"1"
}
]}

If error occurs :

Returns 0;

```
getLimbPerformance ------------------------------------

This method returns the limb performance using the limb ID.

Arguments:

$sessionID – The ID of the session.

$limbID – The limb ID .(If ID=0 return all limbs performances)

Request:

WEBROOT_PATH/kinect.php?action=action=getlimbperformance&limbid=<limb ID>&sid=<session ID>

Response:

If successful :

        {"count":[

                {

                "limb":"11",

                "red":6,

                "green":3,

                "total":9

                },{

                "limb":"6",

                "red":5,

                "green":4,

                "total":9

                }

        ]}

If error occurs :

        Returns 0;
```

Appendix C – Application Manual

This manual aims to expose all the functionalities of the mobile application as well as explain how it works. The application was developed to be a tool to assist therapists with the daily analysis of the patient's session's data.

Only therapist should have access to the data through the mobile application. To be able to use the application therapist should use their credentials (e.g. username, password) to login into the application. The credentials are generated by the system using the therapist's personal data.

Login screen:

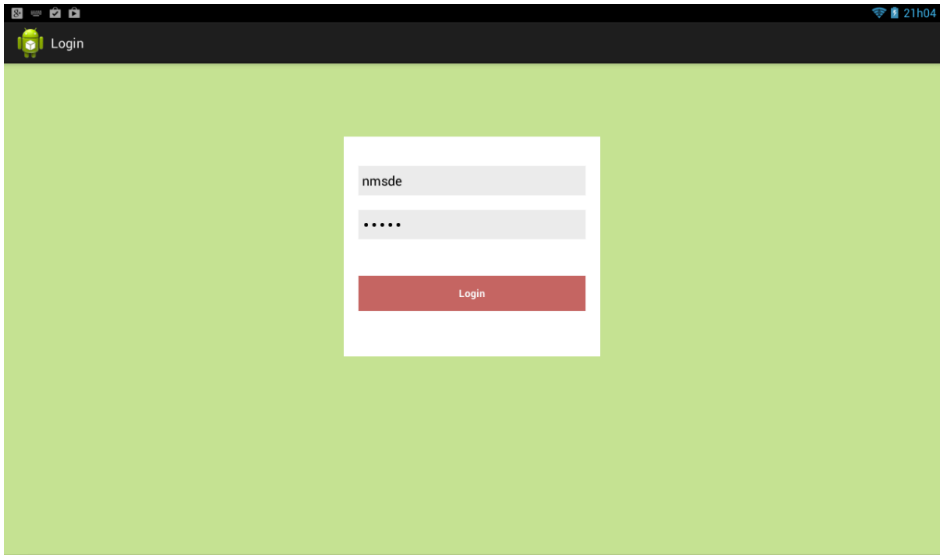


Figure 42 – Mobile application login screen

The therapists should introduce the username into the first field and their password into the second field. To initialize the authentication process the **Login** button should be pressed. If the authentication process goes as expected, the list of patients that is associated with their profile is shown.

Main screen:

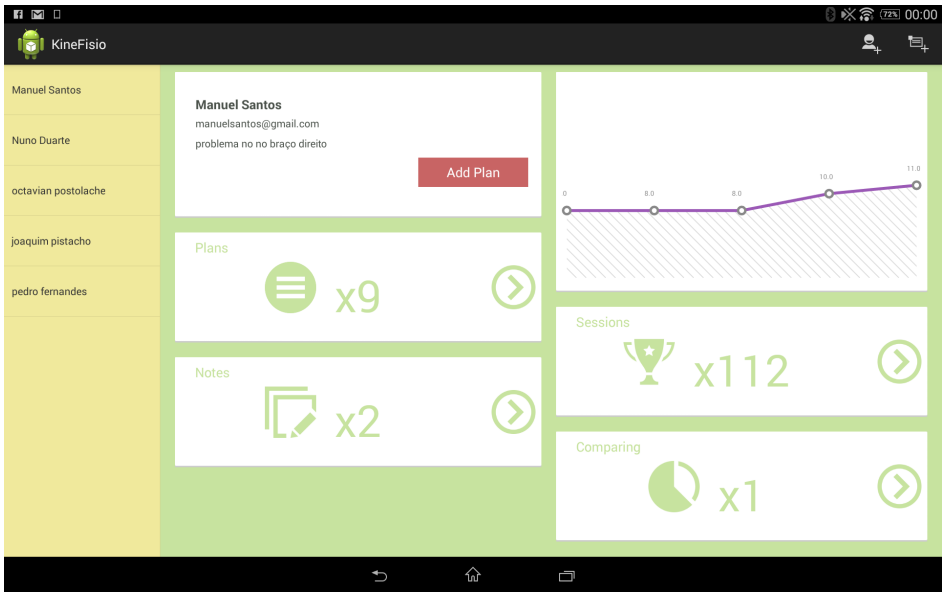
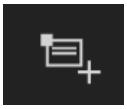


Figure 43 – Mobile application main screen

The main screen is divided in tow areas: the patient's list and the patient's profile. In the upper right corner of the application there is two buttons the first one is used to add patients and the second one is used to add notes regarding to the selected patient.

Add new user



Add new note

When the **Add new user** button is pressed, the add new user screen is showed. This screen allows the therapist to introduce a new patient in the system.

Add user screen:

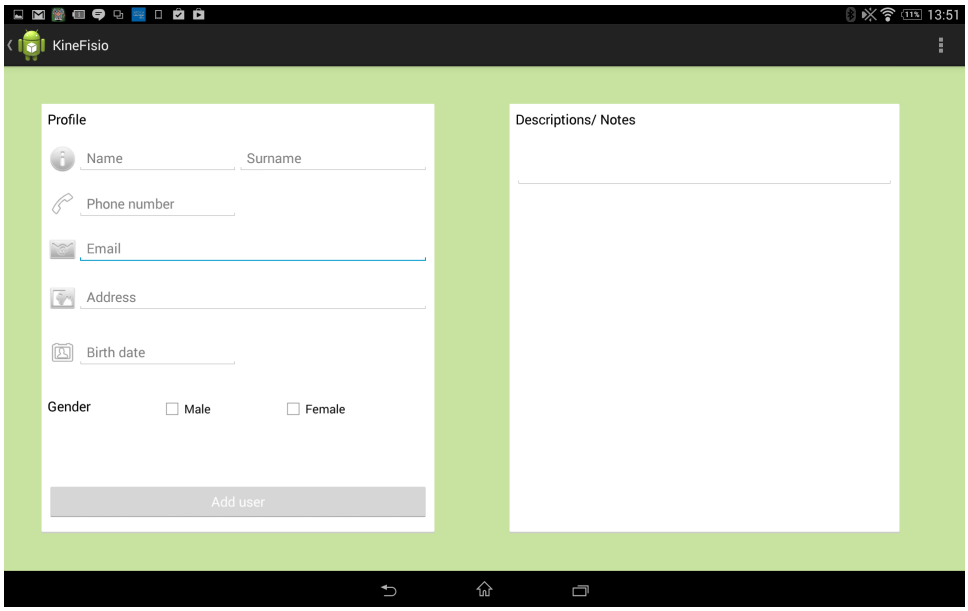


Figure 44 – Add new user screen

This screen allows the therapist to introduce patient's personal data and a description of the problem for which he/she will be treated. To complete the process the user should press the **Add user** button.

If the therapist wants to take a note, he/she should press the **Add new note** button. This button is available on any screen of the application.

Add note screen:

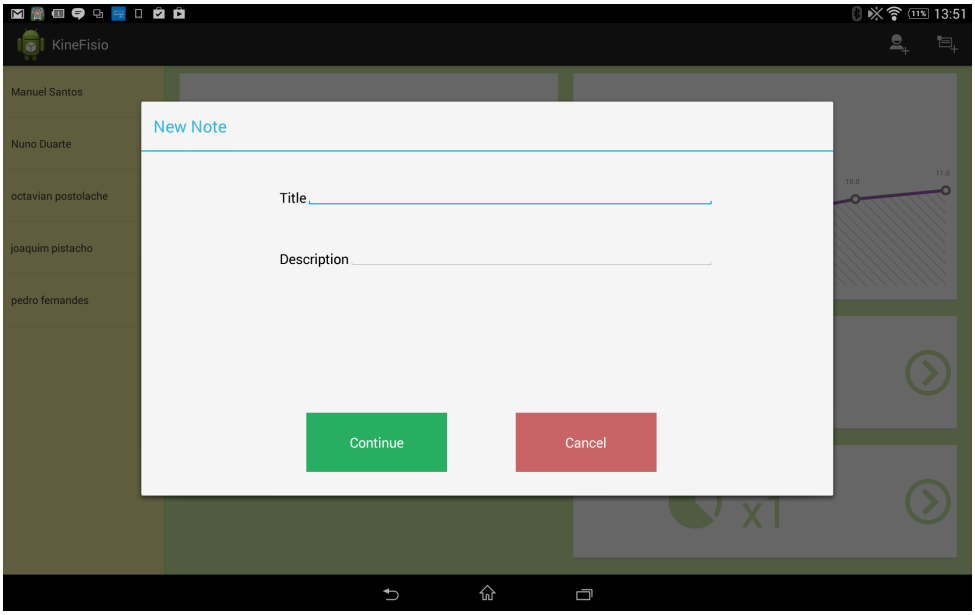


Figure 45 – Add new note screen

To create a note the therapist should introduce the title and the content of the note in the correspondent fields.

When the therapist selects a patient from the list the patient's data is loaded to the profile area. This area shows information about the personal data of the patient, about the progression of the patient using a line chart, give information about the number of notes, plans and sessions executed by the select patient as seen on Figure 43. This area allows the therapist to create a new plan to the patient, using the **Add Plan** button. When this button is pressed the plans setup wizard appears.

Plan setup wizard:

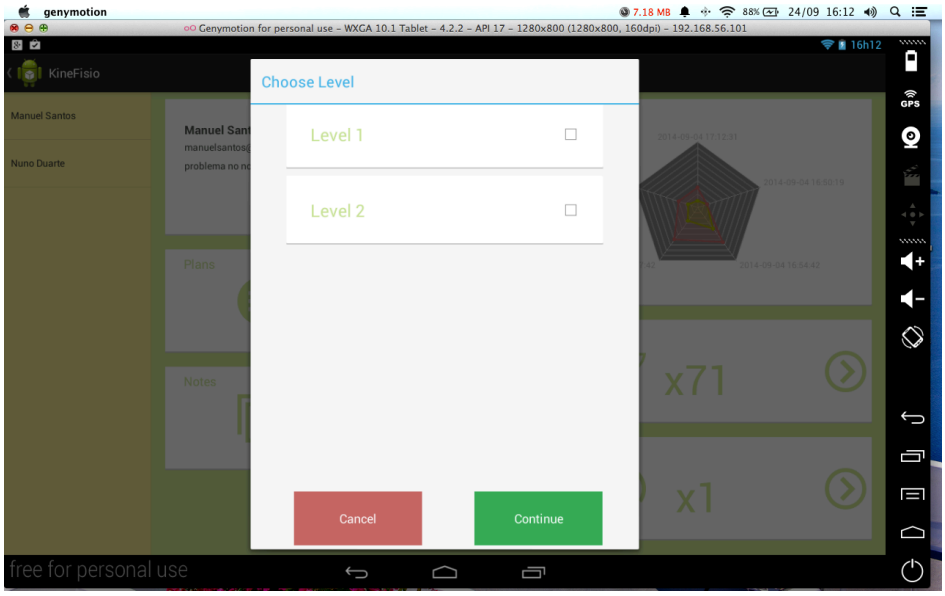


Figure 46 – Setup wizard example

The wizard changes in according to the selected level. The flowchart presented below illustrates the path that the wizard follows when each one of the levels is selected:

Wizard flowchart:

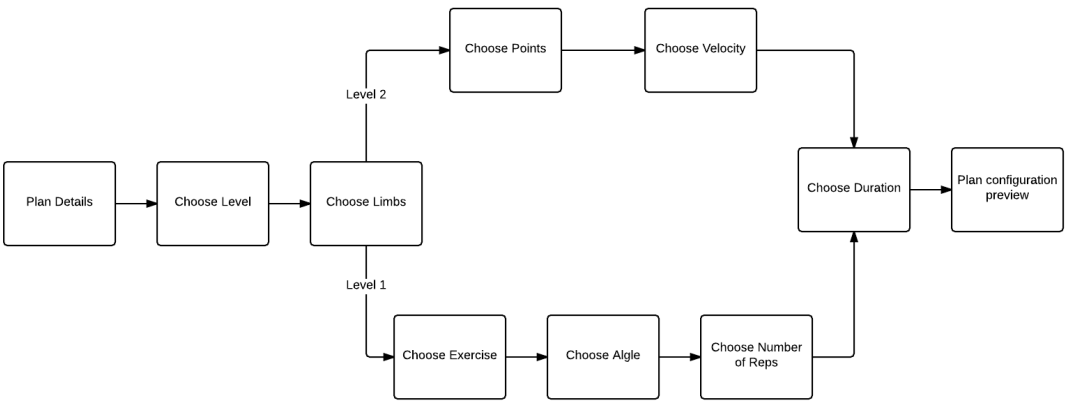


Figure 47- Plans setup wizard sequence diagram

In the profiles area it is possible to see two lists of clickable items:

Access plans list

Access notes list

Access sessions list and individual session analysis

Access the comparison mode

Access plans list:

This view contains a dropdown menu where it is possible to select a plan. When the therapist selects a plan from the list, the details of the selected plan are presented, as seen in Figure 48.

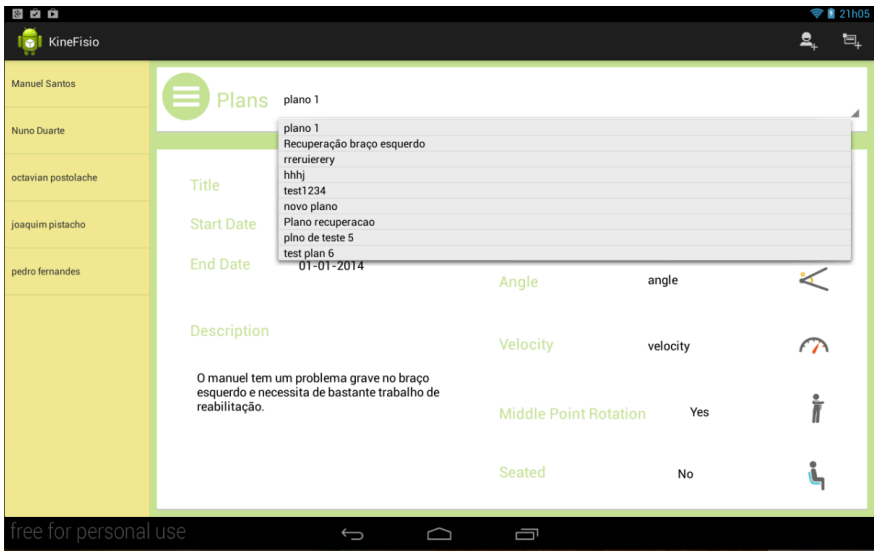


Figure 48 – Plans selectbox

Access notes list:

Similarly to the plan's view, therapists can access to the selected patient's notes and use a dropdown to select a note.

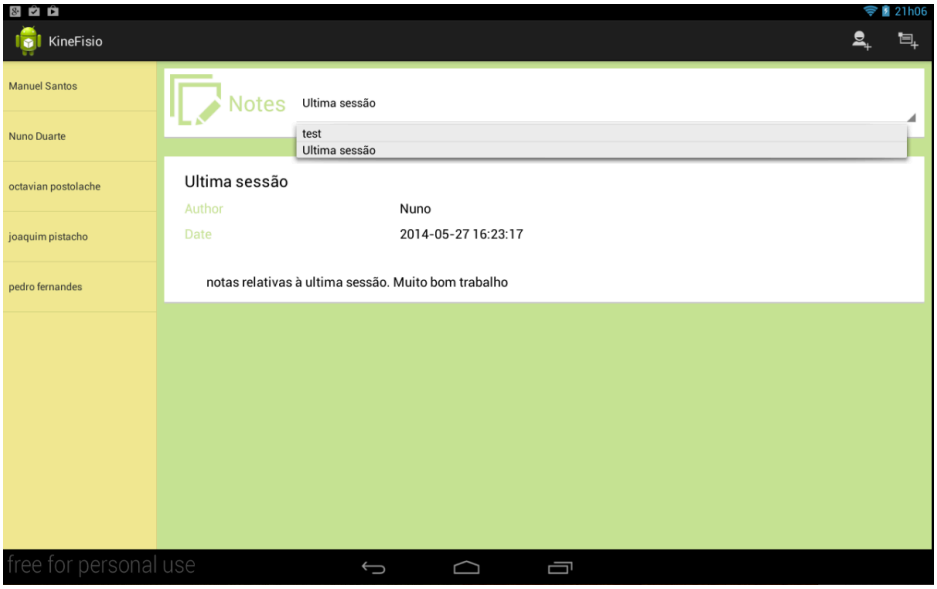


Figure 49 – Notes selectbox

Access sessions list:

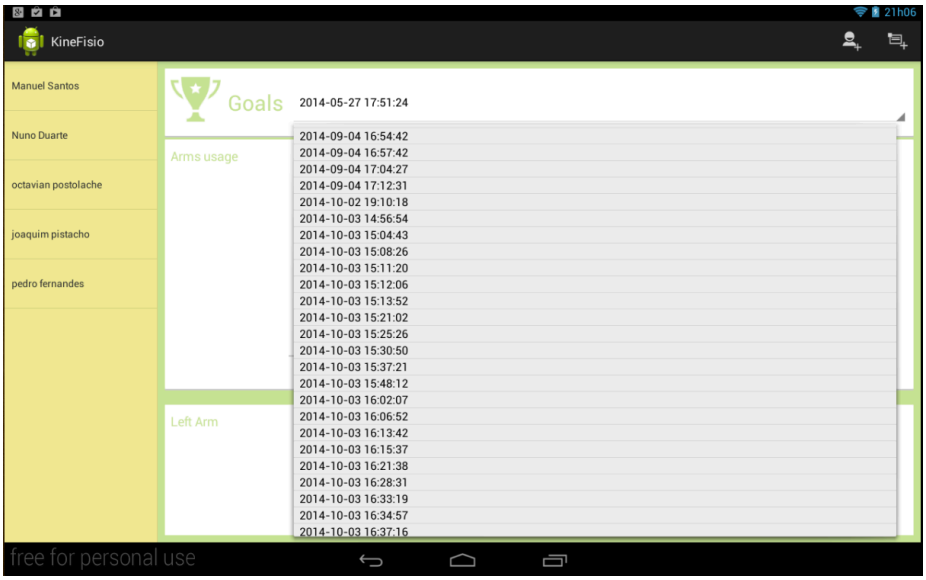


Figure 50 – Sessions selectbox

After selecting a session the therapist can see the statistics associated to the selected session through a set of charts as seen in Figure 51.

Figure 51 – Session's details

Access the comparison mode:

When accessing the comparison mode, the therapist can compare the last five sessions executed by the selected patient as seen in Figure 52.

Figure 52 – Comparison mode

Appendix D – Game Application Manual

The game is compiled in an **.exe** file. To be able to play the game the PC must have a network connection.

1 - To initialize the game run the **.exe** file.

When the game starts the login screen, as seen in Figure 53, will appear.

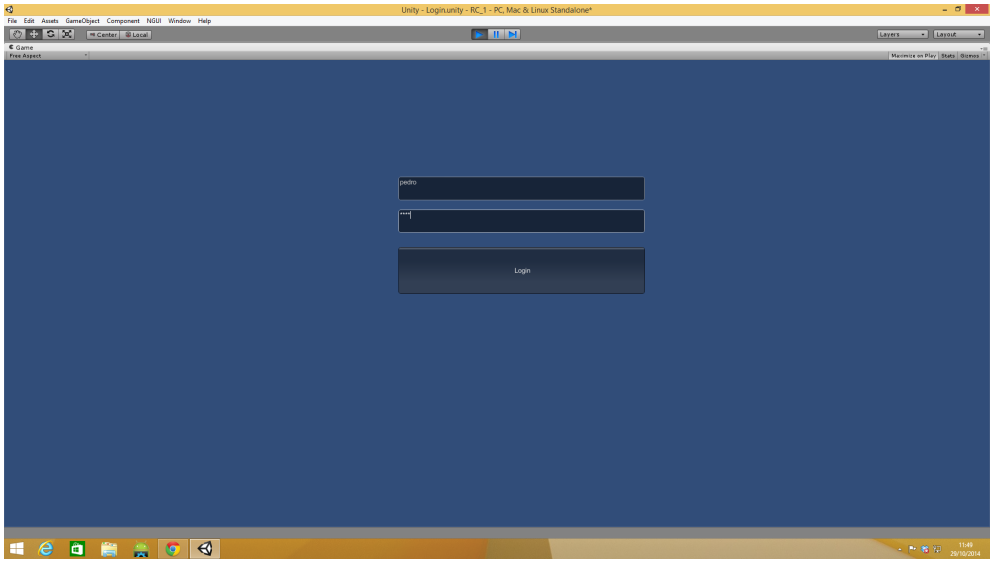


Figure 53 – Game login screen

2 - The patient should introduce his/her credentials and press the **Login** button to start the authentication process. If the authentication process is successful a welcome screen appears and the patient's personal data and the plan configuration are loaded. A description of the session objective is shown as seen in Figure 54.

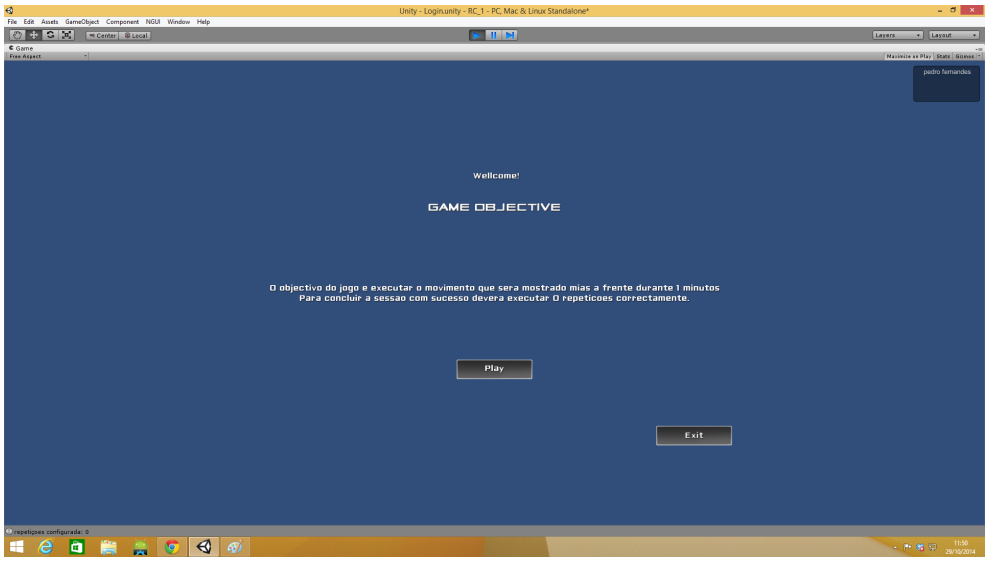


Figure 54 – Game objective preview

3- Press **Play** button to continue or press the **Exit** button to return to the login screen.

When the patient presses the Play button, a screen showing a preview of the movement configured by the therapist with the mobile application is shown. This helps the patient to understand which movement he/she should execute during the session.

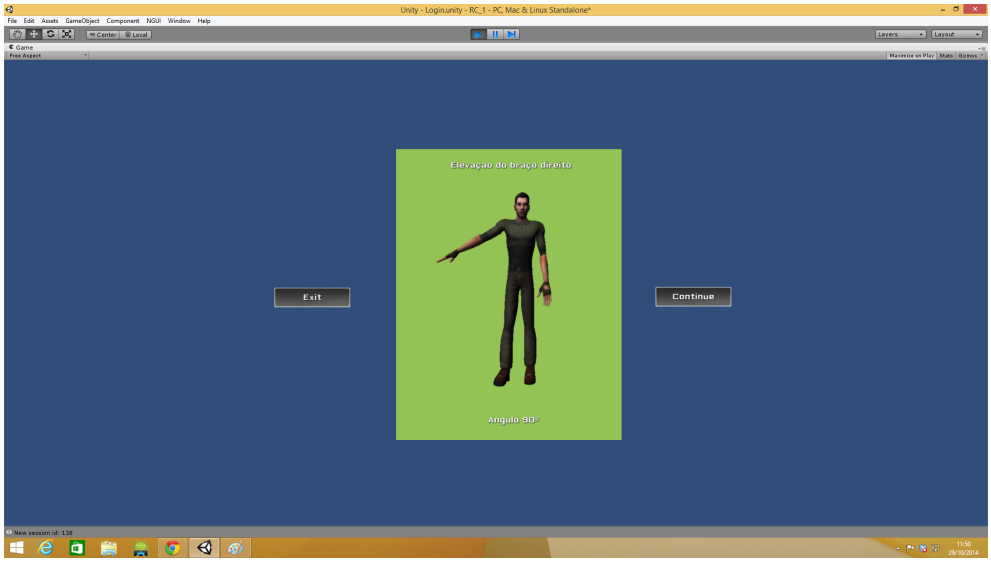


Figure 55 – Preview of the configured movement

4 – Press the **Continue** button to start play the game. After pressing the button the patient should wait a few seconds for the game to begin.

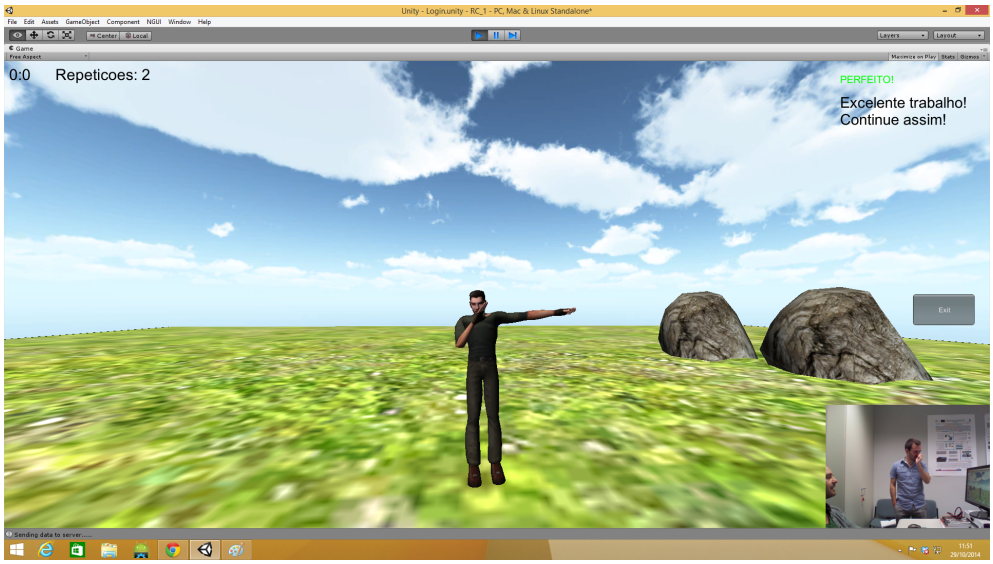


Figure 56 – Game session

To end the game before the defined time, the patient can press the **Exit** button on the right bottom corner of the screen.

Appendix E – Paper

Third International Workshop on Electromagnetic Compatibility and Engineering in Medicine and Biology

EPE 2014

October 2014, 16 -18

Iasi, Romania

KSGphysio – Kinect Serious Game for Physiotherapy

Nuno Duarte
Instituto de Telecomunicações/ISCTE-IUL, Lisboa, Portugal, nuno.msde@gmail.com

Octavian Postolache
Instituto de Telecomunicações/ISCTE-IUL, Lisboa, Portugal, opostolache@lx.it.pt

Jacob Scharcanski
Universidade Federal de Rio Grande de Sul, Brasil, jacobs@inf.ufrgs.br

Abstract— **Serious games fall under a set of applications capable of improving recovery times by increasing the player's engagement. In this paper we focus on the possibility of joining thatcapacity to Microsoft Kinect sensors ability to collect data without the need of additional sensorsand present an application capable of giving proper feedback about the player's behavior during a rehabilitation session. Using an Android OS mobile platform as interface for the collected data,the proposed solution is a prototype that aims to facilitate the analysis of the patient'sprogress during rehabilitation sessions using serious games. Results associated with arm rehabilitation through serious gameare included in the present work.**

Keywords- remote physical rehabilitation, serious games, pervasive computing, virtual reality, personalized medicine

I. INTRODUCTION

In a world where it is increasingly common to integrate virtual environments in real-life situations rehabilitation is no exception. It is increasingly common to use video games to create greater engagement on the user increasing the physiotherapy efficiency. However, because they are adapted games and are not specifically designed for rehabilitation, many therapists prefer to just use them more as a complement to a physical therapy session rather than an integral part of the session. These games are very limited because they are not configurable or accessories are needed to interact with it, which sometimes makes it impossible to use in a rehabilitation session of a more impaired user. Do not allow tracking progress and in most cases there is a lack of feedback to the therapist and to the patients under rehabilitation process [1].

Research suggests that repeated execution of an exercise is sufficient to stimulate the brain to remodel and promote better motor control of the limbs [2]. However, the number of exercises performed during a rehabilitation session is not sufficient for this to happen which suggests the need for patients to practice these exercises at home. However 65% of patients self-reporting being non-adherent or partially adherent to their home programs [3], which in most cases represents a regression of the patient. Adherence embraces 2 elements: being compliant with the frequency of the suggested exercises and carrying out each exercise with the correct biomechanical alignment.

This study aims to produce an application that improves the quality of rehabilitation sessionsin the clinic or at home, while enabling data collection and communication so that we can track the patient physical rehabilitation progress.

II. RELATED WORK

Virtual reality is increasingly used in areas such as rehabilitation through motion-based games. This new generation of tools for rehabilitation has grown up rapidly in the past few years. However, many of these games require wearing a number of sensors attached to the body or use extra material to detect the movements of the patient [4, 5].

In [1], authors propose a rehabilitation tool based on Nintendo Wii complemented with a web-based application that allow motion capture and monitoring exercises to track patient's progression.

Applications based on Kinect sensors natural interaction without the need of additional sensors allow therapists to customize and adjust the physiotherapy training to be performed by the patientand give more freedom of movement.

In [2], after testing with Kinect-based applications in different situations and in patients with different levels of rehabilitation, the authors concluded that the accuracy rate in the detection of motion using a Microsoft Kinect sensor is more than 80%. Therapists rated the technology positively indicating that this kind of tool would reduce their labouring burden and improve rehabilitation efficiency, while patients indicated that this technology has helped to increase their motivation to participate in rehabilitation reducing the time affected by this process.

In [6], authors show their concern about the lack of arguments that validate rigorously the technical performance of Kinect sensor as a rehabilitation tool, although previous works prove the potential of Kinect sensor in this area. They studied the trajectories of the joints at the right hand, right elbow, and right shoulder when performing motor task External Rotation, Scapular Retraction and Shoulder Abduction and compared the results of a Kinect sensor with the results of a OptiTrack that is a marker-based system which requires users to wear reflective markers such that their movements can be tracked by an array of cameras. They also evaluate the timing performance of both systems.

In [7], authors present the results of Kinect applications used in physical rehabilitation tests and foresee that

Kinectsensor technology will be widely applied in medical care fields.

III. SYSTEM DESCRIPTION

The proposed system is divided in three parts, asare presented in Figure 1. The first one is a game that uses the Microsoft Kinect motion sensor to allow data collection during rehabilitation sessions. Every time a patient start his session, a particular game that was previously configured by the therapist who defined some aspects of the game as the speed, the angle that must be performed by a particular limb or the difficulty level of the game given the patient's condition. Using the Microsoft Kinect SDK and a C# wrapper for Unity 3D we can detect automatically the patient joints position and send that information frame-by-frame to the database. Each time a special event occurs, the patient joints positions also are sent to the database. It is understood as special event every time the patient accomplishes a game objective like catching a specific object. In that way we can relate the position of the picked up object and the position of patient's joints and compute physical rehabilitation parameters such as velocity, angles or limbs rotations and compare them with those previously given by therapist through the game configuration interface. The Kinect sensor isconnected to a computer with an Intel core i5-4250U processor, an Intel HD Graphics 5000 and 8G of memory RAM. The computer characteristicsare important to guarantee that the performance of the game is not compromised.The second one is the server, which includes a database, an API and a data processing application to work over the collected data. The API allows every component of the system to communicate with the databasethrough a set of GET and POST methods. The other component held in the server is the data processing application that will translate the frame-by-frame the data collected during the session into metrics that can be easily interpreted by the therapist.

The third component is the mobile application developed for Android OS devicesthat behaves as a client. The application connected to the database and running on the deviceallows the therapist to consult patient's recordsthrough charts, analyze the data collectedfrom previous sessions,do comparisons between session's data, configure new sessions and add reports related to the patient evaluation.

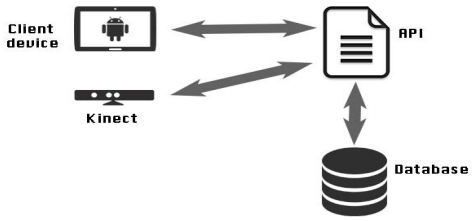


Figure 1 - System architecture

A. *Kinect Sensor*

Kinect is a motion-sensing device that was originally developed for Microsoft's Xbox360 gaming console. The main feature that distinguishes it among others in this gender is that it is not a hand-controlled device, but rather detects your body position, motion and voice. It replaces the controller that was the heart of a gaming device by your body.

The idea of developing the Kinect sensor began to take shape when Microsoft engineers realized that the traditional game controller was the main barrier to making video gaming into a mainstream activity.They quickly realized that the solution was getting a device to track users' bodies as they move.

The core of Microsoft Kinect sensor technology came from a Israeli startup named PrimeSense that figured out how to encode patterns in the light beams and then measuring the changes in those patterns it's possible to give a particularly accurate view of a room [8].

The Kinect sensor(has now outgrown its Xbox roots and is no longer limited to only gaming after Microsoft release Kinect for Windows, a version designed specifically for PC that helps developers to write their own code and develop real-life applications like serious games for rehabilitation purposes with human gestures and body motions.

The Kinect sensor includes a color sensor (Figure 2), an infrared (IR) emitter, a IR depth sensor, a set of microphones and a LED. Additionally has a small motor working as the base that enables the device to be tilted in a horizontal direction.The color camera is responsible for capturing and streaming the color video data. The Kinect color stream supports a speed of 30 frames per second (FPS) at a resolution of 640x480 pixels, and a maximum resolution of 1280x960 pixels at up to 12 FPS.

The IR emitter and the IR depth sensor work together to make things happen. The IR emitter constantly emits a infrared light in a "pseudo-random dot" pattern over everything in front of it, as seen in Figure 3, and the depth sensor reads the reflection of the dotted light in the objects and converts them into depth information by measuring the distance between the sensor and the object from the IR dot was read from [9].

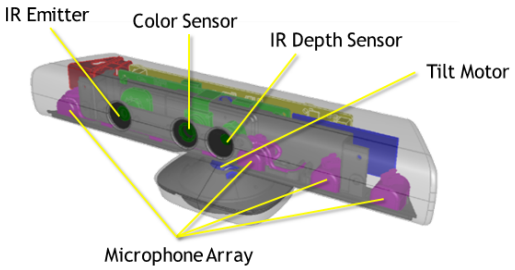


Figure 2 - Microsoft Kinect architecture

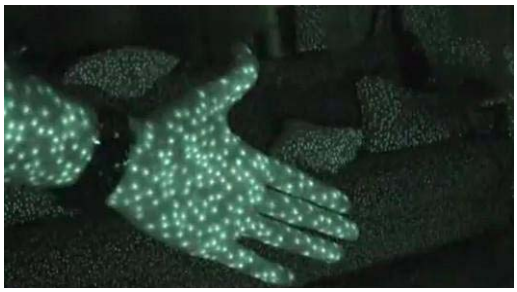


Figure 3 - Pseudo-random dot pattern

B. 3D Game

There are a variety of game engines that can be used to develop 3D games as RAGE (Rockstar Advanced Game Engine) or CryENGINE, this two have been used to create some of the best video games that exist nowadays [10].

To develop the game we used a game engine named Unity 3D that is a powerful rendering engine fully integrated with a complete set of tools and rapid workflows to create interactive 3D and 2D content that has C# as primary language [11]. Unity has a very simple, uncluttered interface for development that allows you to develop games quickly and have a very good integration with Microsoft Kinect sensor. It is also a game engine quite well documented with a strong Asset Store where you can buy scripts, tools and textures to use in the game. Has amazing third party solutions for Audio and Physics and the code is well architectured to reduce the amount of errors done by programmers.

The game environment is based on an orchard composed by rows of trees on both sidesand intends to simulate a harvesting of apples where the user must catch as many apples as possible for a defined period of time, due to this fact the game focuses on the upper limbs of the patient.

There aretwo different kinds of apples, the red ones and the green ones. Different kinds of objectsto choosefrom permits us to define different weights for those objects, whichallows the therapist to understand the cognitive capacities and the patient behavior in situations where he has to take decisions.In this case, the red apples add 100 points to the game score and the green ones add 50 points, as seen in Figure 4, so the patient has to decide if he prefers to catch the green apple that is easier to catch and get only 50 point or if he prefers to catch the red one and reach the goal defined by the therapist faster.

The main difference between this game and the other games mentioned in Related Work is that the parameters related to the rehabilitation process are configurable by the therapist. Using the mobile interface (D) the therapist can define rehabilitation parameters such as the speed, the angle that must be performed by a particular limb, session duration, if the user should execute rotation of the body or not, the difficulty level of the game given the patient's condition and the minimum number of points to do at a session setting up a goal for the patient to commit with.This last parameter is used to compare the performance of the patient between sessions. Every time the patient picks up and object some points are added to the patient's score, in the end of each session the score should be equal or bigger than the minimum defined by the therapist, a lower score may portray a regression of patient's condition.All of these aspects will contribute to help us mapping the patient's progress during the rehab time.

C. Database model

For database we used a relational database management system (RDBMS) named MySQL [12] that has no GUI tools to administer databases or manage data contained within the databases. For that reason a front-end tool named MySQL Workbench [14] was used to

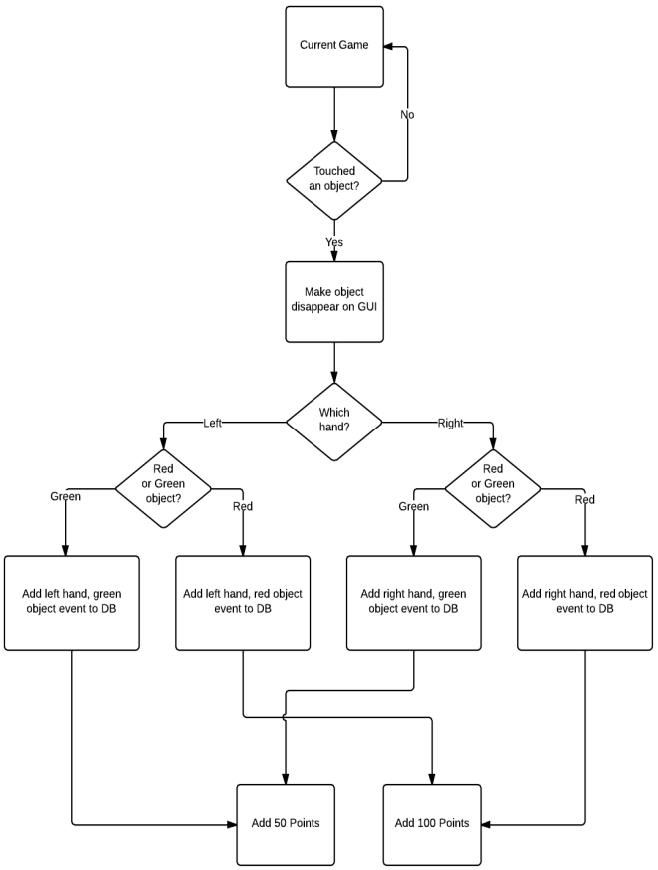


Figure 4 - Pick up object flowchart

enable us to graphically create and manage the tables of our database.

We can divide de data stored in the database in two main data types. The first type is the one that can be added to database using the mobile application interface: patient's personal data, notes written by the therapist, prescribed plans and game configurations. The second type is the data collected by the Kinect sensor like limbs positions.

The system users identificationsare stored in the database grouped in two types:therapists and patients. Each therapist has a set of patients that he can manage through the mobile application.Each patient can have a set of plans associated to his profile that are defined by their therapist. These plans have an associated configuration defined by the therapist during the creation the plan that will be used in the construction of the 3D game scene when the user starts a session in the game. Each plan consists in a set of sessions organized in time, which in turn consist of a collection of scenes.

Each frame collected by Kinect sensor represents a scene and consists in a group of three-dimensional vectors (x, y, z) corresponding to the position of each member during that scene. In addition to the data collected by the sensor frame by frame, there are special events like picking up an apple, that are saved so that we can map the evolution of the patient,by

comparing the number of picked up apples between a set of sessions performed by the patient. If the number of picked up apples increase, that could mean that the patient is improving.

D. Physiotherapist's Mobile Interface

The Android OS is one of the most used operating systems on mobile devices by virtue of being open-source and still growing. Well positioned on the market Android achieved the 81.1% market share in the first quarter of 2014 [13].The mobile application was designed to permit the interaction between physiotherapist and the data provided the client and stored on the serverside. It is important that the application can display the data in a clear, synthesized and objective way so that the therapist can easily understand them. To achieve that we use a set of different types of charts as line charts, pie charts, bar charts and radar charts to display the data graphically.To create the charts we useHoloGraphLibrarylibrary was used [15] that is simple to use and to integrate with the application. It uses Android Canvas, which works as an interface to the actual surface upon which the graphics will be draw. It's the best choice when the application needs to regularly re-draw itself.

The communication between the mobile application deployed in Android device and the server is done using a networking library named Volley [16] that speeds up and facilitates that communication by allowing asynchronous HTTP requests and dealing with cache.

The interface splits the data in two types: data concerning to the game goals and data concerning to the limbs behave. The first type is important to understand the evolution of the cognitive capacities of the patient by showing to therapist statistics like the percentage of green and red apples picked up during a session, comparison between the number of picked up apples in the last 5 sessions, score progression, percentage of sessions where the patient achieved the minimum score defined. Data are presented in a generic way by making an overview of the patient's progress, but the therapist can also select a specific session and analyse every session individually. The second type represents the behaviour of the patient's limbs, using the data collected by the Kinect sensor we can compare the limbs performance, understand which limb is more frequently used to pick up apples, the progression of each member, correctly execution of an angle, the average velocity of the session and distinguish if the patient did more points with his left hand or his right hand.

The mobile software, which flowchart is presented in Figure 5, permits to the therapist to execute other tasks as add new patients, to create plans related to a patient's profile and configure them, this configuration will allow the system to adapt the game according to the patient's needs, add general notes or notes related to a specific session.

IV. RESULTS AND DISCUSSIONS

In this section, are presented the results associated with the tests made using the Kinect based systemand the developed rehabilitation game for different users. Was tested the capacity of the designed and implemented system to collect data related to range, as it seen in Figures 6 and 7, and velocity, as seen in Figures 8 and 9,while patientsplayed the game. The represented data was normalized.The developed mobile interface is also presented with some results concerning the EHR and physiotherapy training data visualization.

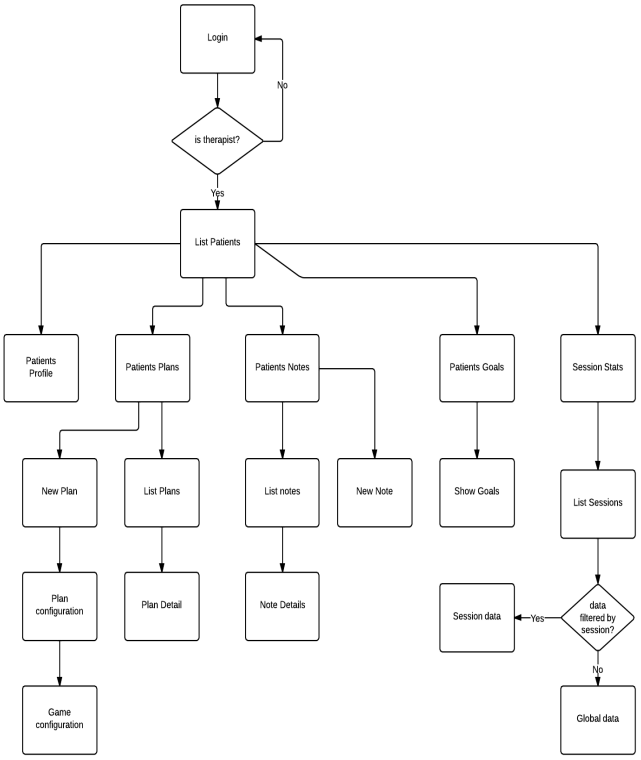


Figure 5 - Mobile application flowchart

V. RESULTS AND DISCUSSIONS

In this section, are presented the results associated with the tests made using the Kinect based systemand the developed rehabilitation game for different users. Was tested the capacity of the designed and implemented system to collect data related to range, as it seen in Figures 6 and 7, and velocity, as seen in Figures 8 and 9,while patientsplayed the game. The represented data was normalized.The developed mobile interface is also presented with some results concerning the EHR and physiotherapy training data visualization.

We carried out this study with two male participants. Each participant played the pick up apples game during two minutes. We choose eighty-seven seconds of the collected data to evaluate the performance of the system regarding to patient's hands range and velocity.To perform the test, each participant was standing up three meters away in front of the Kinect sensor.

Figures 6 and 7 represents the ranges associated to left hand and right hand motion performed during a rehabilitation session showing that Kinect sensor has the capacity to collect sufficient data that can be used to track the evolution of the patient's hands range. Acquiring data from consecutive sessions performed by the same patient may help to analyze the values of the measured ranges during different physiotherapy sessions of the range and evaluate the rehabilitation evolution process.

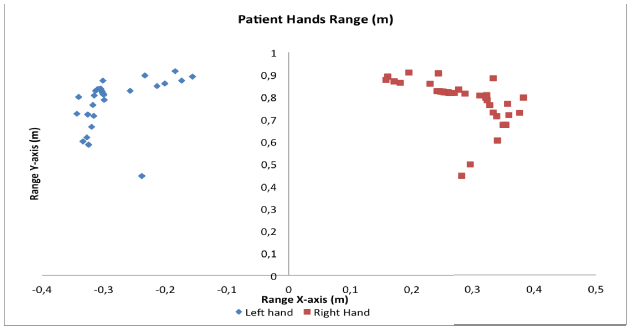


Figure 6–Left Hand/ Right Hand range test patient 1

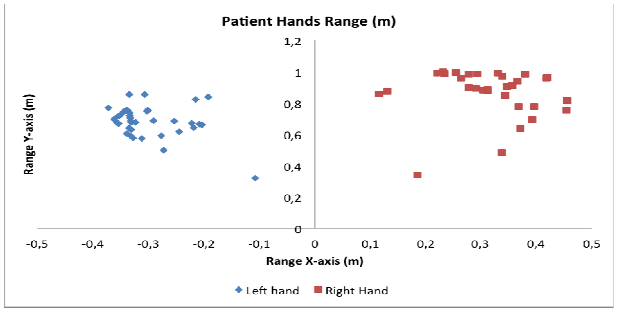
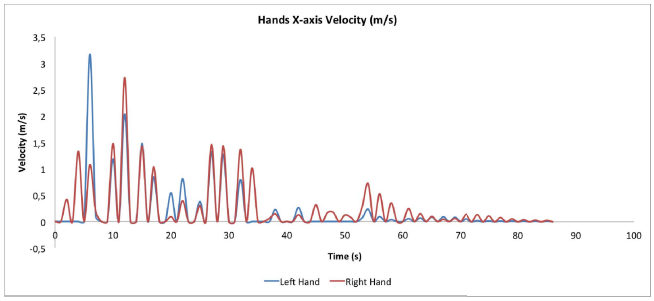
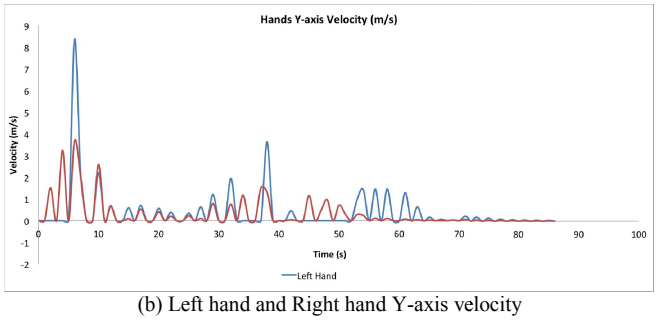


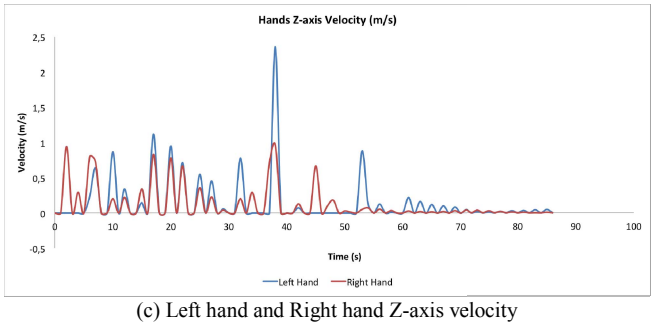
Figure 7 - Left Hand/ Right Hand range test patient2



(a) Left hand and Right hand X-axis velocity

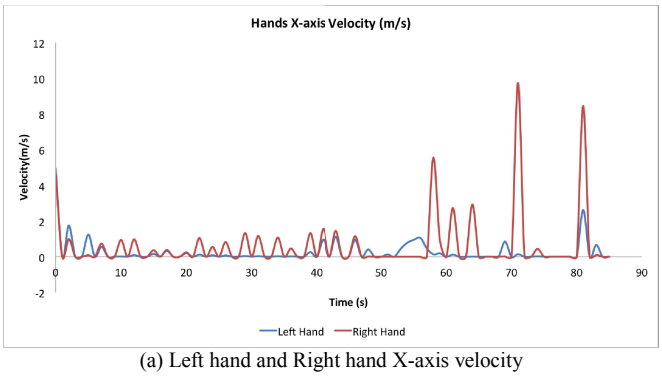


(b) Left hand and Right hand Y-axis velocity

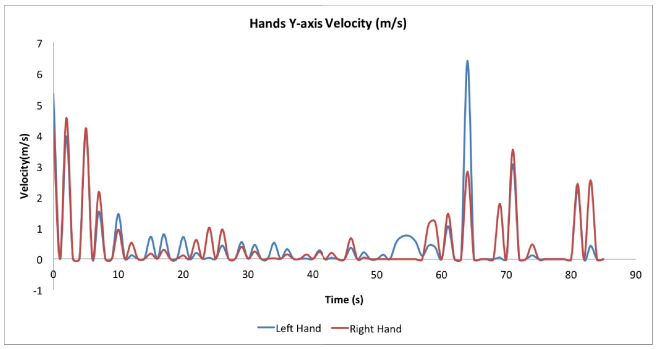


(c) Left hand and Right hand Z-axis velocity

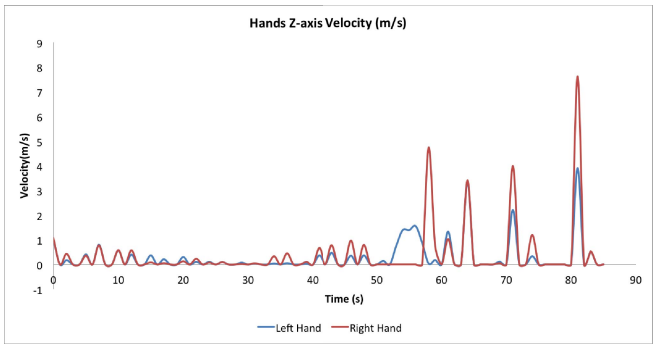
Figure 8 - X-axis, Y-axis, Z-axis patient 1 hands velocity performing a game session



(a) Left hand and Right hand X-axis velocity



(b) Left and Right hand Y-axis velocity



(c) Left hand and Right hand Z-axis velocity

Figure 9 – X-axis, Y-axis, Z-axis patient 2 hands velocity performing a game session

In a similar manner, using the Kinect sensor we can calculate the velocity of the left and right hands regarding to X-axis, Y-axis and Z-axis. Given the velocity values in different axis may help to understand the mobility of a patient's arm in different orientations.

Using the mobile application the therapist has the access to statistics regarding patient's rehabilitation sessions. Figure 10 and 11 illustrate some of that statistics, in Figure 10 we can see the comparison between the number of objects picked up by the patient in the last five sessions sorted by colors, so the the rapist can compare the patient performance and cognitive capacity. In Figure 11 we can see the user interface showing an overview of the patient's progression in the last five sessions, in terms of points obtained in each session, and the number of red and green apples caught.

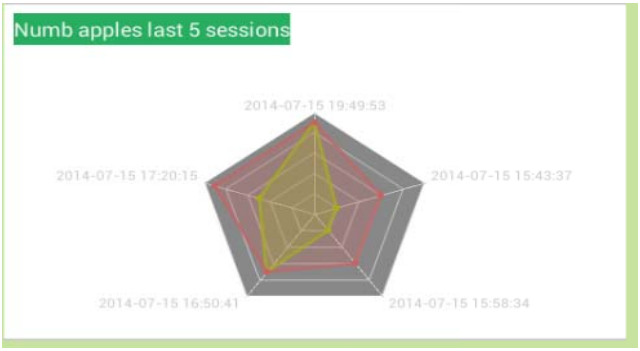


Figure 10–Mobile application graphic comparing the number of red apples and green apples picked up between patient's last 5 sessions

Figure 11 - Mobile application showing patient's progression in last 5 sessions

VI. CONCLUSIONS

Microsoft Kinect sensor is an appropriate tool for data collection because it enables an appropriate analysis of patient's behaviour during a rehabilitation session.The advantagesof combining the Kinect sensor with the rendering engine Unity3D and mobile applications to create 3D serious games capable of being configurable and adapted to the patient's needs are undeniable. Using Unity3D allows us to replicate real life situations, which facilitates the transition between virtual and real world. The mobile application brings mobility to data analysis allowing the therapist to see relevant statistics of a patient session, enables a single therapist to monitor several patients at the same time and allows him to configure specific parameters as velocity or angles of a session according to the patient's needs. Both, game and mobile applicationwere presented during demonstration workshops and proved to have a good acceptance in the therapist community.

The ongoing work is focused on an evaluation of the effectiveness and usefulness of the system, using two control groups: a group of therapists to test the mobile application and a group of patients to test the game represents.

ACKNOWLEDGMENT

This work was supported by the Instituto de Telecomunicações and Fundação para a Ciência e Tecnologia and the project "EHR-Physio: Electronic Health Records-Needs, Requirements and Barriers in Physiotherapy", PTDC/DTP-DFS/1661/2012 a specialacknowledgement is granted.

REFERENCES

[1] Fraser Anderson, Michelle Annett, Walter F. Bischo, Lean on Wii: Physical Rehabilitation With Virtual Reality and Wii Peripherals, Stud Health Technol Inform. 2010;154:229-34.

[2] Jun-Da Huang , Kinerehab: A Kinect-based System for Physical Rehabilitation — A Pilot Study for Young Adults with Motor Disabilities 2011, Research in Developmental DisabilitiesVolume 32, Issue 6, November–December 2011, Pages 2566–2570.

[3] Bassett.H.S, The Assessment of Patient Adherence to Physiotherapy Rehabilitation. New Zealand Journal of Physiotherapy 2003; 31(2): 60-66

[4] H. Sugarman, A. Burstin, A. Weisel-Eichler, and R Brown. Use of the Wii Fit System for the treatment of balance problems in the elderly: A case report. Virtual Rehabilitation, Pages 111-116, 2009.

[5] J. Deutsch, D. Robbins, J. Morrison, and P. Guarrera-Bowlby. Wii-based compared to standard of care balance and mobility rehabilitation for two individuals post-stroke. Virtual Rehabilitation, Pages 117- 120, 2009.

[6] Chien-Yen Chang Lange B. , Mi Zhang , Koenig S. , Requejo P. , Noom Somboon , Sawchuk A.A. , Rizzo A.A. Towards Pervasive Physical Rehabilitation Using Microsoft Kinect, Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2012 6th International Conference , Pages 159 – 162 , 2012.

[7] NaofumiKitsunezaki, Eijiro Adachi, Takashi Masuda , Jun-ichiMizusawa : KINECT Applications for The Physical Rehabilitation, 2013, Medical Measurements and Applications Proceedings (MeMeA), 2013 IEEE International Symposium, Pages 294 – 299, 4-5 May 2014.

[8] The story behind Microsoft's hot selling kinect. Website: http://www.businessinsider.com/the-story-behind-microsofts-hot-selling-kinect-2011-1

[9] Abhijit Jana. Kinect for Windows SDK Programming Guide, December 2012

[10] The 10 best game engines of this generation. Website: http://uk.ign.com/articles/2009/07/15/the-10-best-game-engines-of-this-generation

[11] Unity 3D game engine. Website: http://unity3d.com

[12] MySQL. Website: http://www.mysql.com/

[13] MySQL Workbench. Website: http://www.mysql.com/products/workbench/

[14] Smartphone OS market share, IDC Analyze the future, Website: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[15] HoloGraphLibrary, Website: https://github.com/Androguide/HoloGraphLibrary

[16] Volley library Website: https://developers.google.com/events/io/sessions/325304728