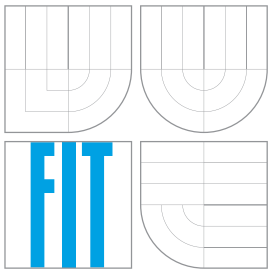
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ZPRACOVÁNÍ MEDIÁLNÍCH DAT

MEDIA DATA PROCESSING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL KRČA

VEDOUCÍ PRÁCE
SUPERVISOR

PAVEL ZEMČÍK, prof. Dr. Ing.

BRNO 2014

Abstrakt

Tato práce popisuje proces tvorby multi-platformního interaktivního systému a demonstruje jeho použitelnost na prototypu aplikace simulující rehabilitaci. Je zde zahrnut SW návrh, 3D programování a produkce animací za použití hloubkového senzoru Kinect. Jádro aplikace bylo vytvořeno v herním enginu Unity 3D a bylo otestováno na Linux Ubuntu, Windows 7, mobilním zařízení Android a Unity Web Playeru.

První polovina tohoto dokumentu obsahuje můj průzkum nejmodernějších nástrojů a zdrojů vhodných pro realizaci interaktivní multi-platformní aplikace, pracující s 3D obsahem. Nejprve jsou popsány tři herní enginy (Unity 3D, UDK, Unreal Engine 4), jakožto hlavní vývojová prostředí vhodná k realizaci aplikace. Poté jsou popsány možnosti ukládání dat obecné pro všechny zmíněné enginy - integrovaná úložiště dat, databázové systémy a serializace. Nakonec jsou v práci popsány výhody a nevýhody různých přístupů, jak vytvořit 3D animace. Je zde naznačeno použití 3D softwaru určeného k manuální produkci, ale také různé možnosti snímání pohybu Motion Capture.

Druhá část dokumentu popisuje mé zhodnocení informací popsaných v předchozí části a rozhodnutí, které prostředky jsem použil k vytvoření prototypu. Také je zde zkonkretizováno zadání a formulovány základní parametry aplikace. Dále je popsán SW návrh, produkce a zpracování 3D animací a samotná implementace. Nakonec je výsledný prototyp zhodnocen a naznačena rozšíření, které je možné aplikovat v budoucnosti.

Co se týče využití samotné aplikace, tato práce předvádí mé nápady, jak mohou být informační technologie využity v oboru fyzioterapie a zdravotnictví obecně. Výsledkem nemělo být nahrazení fyzioterapeutů, nýbrž poskytnutí pomoci jim a jejich pacientům. Tento prototyp je příkladem, který nabízí 3D vizualizaci cvičení a lidského pohybového systému. 3D animace, ukazující, jak by měl být každý cvik správně proveden, mimo jiné, mohou učinit rehabilitace příjemnější a zároveň účinnější.

Abstract

This work describes a process of creating a cross-platform interactive system and it demonstrates usability on prototype application that simulates a rehabilitation session. The application includes 3D visualization possibilities - interactive human muscular system and exercises with 3D animations. The work included SW design, 3D programming and production of animations using Kinect sensor. The core of this prototype was created in Unity 3D game engine. It was deployed and tested on Linux Ubuntu, Windows 7, Android mobile device and Unity Web Player. This work as well manifests my ideas how Information Technology can be used in field of physical therapy.

Klíčová slova

Interaktivní mediální systémy, Unity 3D, 3D programování, 3D animace, Kinect senzor, fyzioterapie, informační technologie ve zdravotnictví

Keywords

Interactive media systems, Unity 3D, 3D programming, 3D animations, Kinect sensor, Physical therapy, Healthcare Information Technology

Citace

Michal Krča: Media Data Processing, bakalářská práce, Brno, FIT VUT v Brně, 2014

Media Data Processing

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have faithfully cited all sources used in the thesis.

.....
Michal Krča
May 12, 2014

Acknowledgments

First of all I would like to thank my Finnish supervisor Pekka Nisula for his open-minded and friendly approach which allowed me to work on something I care about and in way I am comfortable with.

Next I would like to thank the representative of international physical therapy students and above all my friend Marius Radu for his immense help with Kinect recording.

Finally I want to thank my Czech supervisor Pavel Zemčík for his help related to formal form of this document.

Contents

Chapter 1

Introduction

This work describes a process of creating a system running on computers and mobile devices while using 3D content. It includes a research on the most powerful tools and ways how it is usually done as well as my evaluation and decision how to create such a system. The important part of my work was also to obtain and process suitable 3D models and animations. The theme of my system is related to health care and it should manifest how Information Technology could be used to improve human health. Therefore it should be not only functional application, but also a demonstration of real usage in future.

The result of my work should be a prototype of cross-platform interactive media system working with 3D models and animations. I decided that it should simulate a rehabilitation session related to weak or shortened muscles and be as visual and as user-friendly as possible. This prototype should be runnable on mobile devices, standalone computers and even on some of web browsers. The emphasis was also put on rapid development which in my case meant to make a decision which available resources and tools should I use to create the prototype. I decided to use the most modern and widely used tools and at the same time find a balance between the rapid progress and the limited budget. The prototype should be also a demonstration of potentially commercial project and it should include sample exercises.

This document was written abroad via the Erasmus program at the Oulu University of Applied Sciences (OAMK) in Finland. The OAMK assigned me a list of possible project works to choose from and provided financial and specialist support for completing my task. The topic of rehabilitation, well-being and health care itself caught my attention as well as the open-minded attitude of the school which allowed me to use the most modern tools and resources available including special 3D virtual laboratory or Kinect depth-sensor. Therefore I used the project work as my bachelor thesis.

Within three following chapters I want to point out and briefly describe tools and ways how cross-platform interactive 3D content applications are usually done or can be done. I do not want to point out every single possible way or tool which can be used to create the application. I want to show only several of the best ways and tools widely used that are suitable for the main goals of my project work instead. After that a chapter that consists of my conclusions and decisions what and how I was going to do follows. The next two chapters come after and the main content of these is the description of my actual work. At the end of the document, there is a conclusion of my entire work.

Chapter 2

Interactive 3D content software

This chapter initiates a description of my research on the most modern tools and resources suitable for realization of interactive cross-platform 3D content software. I am going to describe here core software designed for 3D content - which means support of various manipulations with 3D models, animations and other features important for my particular project work.

These days (2014) the most usual tools used for 3D interactive content applications are 3D game engines. Naturally, there is plenty of them, but I decided to write down only the most powerful ones relevant for my thesis which met the main requirements of my task. To each I wrote down only the information related to my project work. The final evaluation of these tools will be made in chapter 5.

2.1 Unity 3D

Unity 3D is a development engine that provides rich functionality suitable for interactive 3D and 2D content and quick development which was one of my key targets. But the most important was the Unity's efficient multi-platform publishing. [31]

2.1.1 Licensing

There is a free version of *Unity* which is unlike the various professional licenses limited by many features concerning graphics, animations, deployment and others. The interesting fact is that even the free version can be used for commercial purposes. It is however limited by maximum turnover US$100,000 (2014) in the fiscal year. [31]

2.1.2 Multi-platform

In the *Unity engine*, it is possible to write a code working across all the major platforms. It provides an export and publishing the project on the following platforms: Mac OSX App, Windows Executable, Windows Store, Linux desktop, Web Browsers, via Unity Web Player, iPhone, iPad, Android phones and tablets, Windows Phone 8, Blackberry 10, WiI U, PS3 and Xbox 360. The Unity Web Player is a plug-in needed to run Unity applications in browsers. So far this plugin is not supported on Linux platform. [31]

2.1.3 Programming Language

Unity allows developers to choose from three programming languages - *UnityScript* (language resembling Javascript by syntax but with different semantics), C# or Boo (objected-oriented language with simillar syntax as Python). It is also possible to use all of them, make some scripts in C#, and the others in Boo or UnityScript. [31]

2.1.4 3D content

Unity enables to import animations and models from almost any 3D application. Moreover when the model or animation is saved, Unity immediately re-imports the asset which was updated and afterward it applies changes across the project. It has also some integrated features, such as lighting, basic 3D models and others. *Unity* uses the *Nvidia PhysX engine* for its physics which means it has realistic and high-performance support. The free version, however, lacks of advanced graphic features such as real time shadows. Needless to say, there is significant difference between graphics quality in free version and pro version.

Figure 2.1: Gloria Victis - game created in Unity[1].

2.1.5 Animations

Unity uses 3D animation-system called *Mecanim*. This system supports skeletal animation - setup of animations on humanoid characters. The Mecanim enables management of complex interactions between animations with a visual programming tool. Among the other things, it provides a possibility to animate different body parts with different logic. [30]

[1]Source: http://gloriavictisgame.com/

2.1.6 Development

The high speed of the development is the result of intuitive user interface and plenty of useful features. The considerable impact on this speed also has the fact that the engine is widely used (more than 1.2 million users in the summer of 2013), it has sorely active user community which provides sharing information and therefore faster solution of possible problems. [2]

2.2 Unreal Development Kit

Unreal Development Kit (UDK) is a free edition of award-winning and very popular 3D game engine *Unreal Engine 3* created by *Epic*. [6] The *Unreal Engine* was used not only in many successful games, but also in other areas such as simulation. [24] Just like *Unity*, in *UDK* it is possible to write the same code which works across several platforms.

2.2.1 Licensing

Unlike *Unity*, for any commercial use of *UDK* the developer need to purchase a license ($99) and then pay fee of 25% of his earnings after the first $50,000 income. The full source licenses naturally provides wider functionality and platform support.

2.2.2 Multi-platform

The free version of the engine provides an export on the following platforms: Mac OSX App, Windows Executable, iPhone, iPad and iPod. The full source licenses also supports Android phones, Playstation 3, PlayStation Vita, Xbox 360, WiI U and Windows RT / Windows 8.[8]

It is also possible to visualize the application on-line in web browsers supporting *Adobe Flash* platform.

2.2.3 Programming Language

In spite of the fact that the core of the engine is written in C++, the creators managed to develop their own language, called *UnrealScript*. The *UnrealScript* is objected-oriented language very similar to Java and influenced by C++. [32]

2.2.4 3D content

This engine is using *Nvidia PhysX engine* as well, but unlike *Unity*, a rich set of tools for level design is directly part of the *UDK*. Entire graphics is at higher level, it includes besides other things powerful features such as real time shadows, render to texture and various effects. Naturally, the system requirements are higher.

Although the importing is not so easy in *UDK*. It does not support as many file types as *Unity* and the refreshing of assets is not automatic. The next difference is that while importing it creates its own package.

2.2.5 Animations

UDK uses also very powerful visual skeletal animation system. It offers many features such as *FaceFX* facial animation that enables to produce realistic facial animation from audio

files, inverse kinematics, morphing, state-of-the-art animation and others. The animation is controlled by AnimTree - a tree of animation nodes. [7] [5]

2.2.6 Development

Unlike the full version, there is no natural development environment in *UDK* ready to use immediately after installation of the engine. The developer therefore has to find a proper environment which can collaborate with *UnrealScript*. [32]

The *UDK* however has a very powerful tool called *Kismet* which is a visual scripting system allowing people without any knowledge of programming to design and modify the game. [6]

The engine also has a really large base of users (more than 2.25 million unique installs in the autumn of 2013) and there is plenty of tutorials available. [14]

This free version has been widely used for 5 years, because the the full license was accessible only to professional game development companies. And despite being limited, it provided powerful tools and features to create complex projects. The *UDK* had been also updated frequently. But at the end of March 2014, the *Epic* released a new version of Unreal Engine and significantly changed its licensing model.

Figure 2.2: Mass Effect - popular game using Unreal Engine[2].

2.3 Unreal Engine 4

Even though the freshly released version of *Unreal Engine* is not for free, for many independent 3D interactive application developers this step meant literally a „revolution“. Rich support of various platforms was uppermost interesting for my project work, but also highly probable rapid development.

[2]Source: http://masseffect.bioware.com/

2.3.1 Licensing

There is no free version of Unreal Engine 4, only the full version and it is available from now on even for individuals. It includes the full features, tools and the entire source code. To use this engine it is obligatory to subscribe for $19 a month and to pay 5% fee of the earnings. It is also possible for the same price to install and use the unlimited engine at any academic institutions. [8]

2.3.2 Multi-platform

As mentioned earlier, the engine release is completely new and naturally not tweaked to perfection yet. At its actual state, it is possible to export the application on the same platforms as the full version of *Unreal Engine 3*. There is, however, ongoing work on the other platforms, such as Linux, HTML5 and Oculus VR. The support of console applications should be also possible, but it requires custom licensing offered by *Epic* company. [8]

2.3.3 Programming Language

Unlike the *UDK*, the main programming language of *UE4* is C++ and all the engine is highly optimized. For designers there is also possibility to use *Blueprint visual scripting*. [8]

2.3.4 3D content

The 3D content tools integrated in UE4 are naturally even more advanced than in its predecessor. One of the main changes was to make the scenes much more realistic, for instance, for the first time it enables real-time global illumination.

The importing is now slightly simpler and more intuitive, but it is still not fully automatic, as it is in *Unity*. [8]

2.3.5 Development

The *Blueprint visual scripting* allows to quickly prototype and build playable content without writing any code. The *Kismet* evolved to much more powerful system enabling more extensive changes. Naturally, it makes development even more rapid. The radical removal of previous *UnrealScript*, however, can have a significant impact on the large *UDK* community. [8]

2.3.6 Animations

There is a new *Persona* animation system which is built on top of the *Blueprint visual scripting* system. It is designed to edit animation Blueprints, skeletons, skeletal meshes and more. It enables to preview animation sequences and morph targets during setup, and also to set up montages and animation blend spaces. It is also possible to modify physics and collision properties for skeletal mesh actors. The advanced level of this animation system results in creation of more realistic animations than before. [8]

Chapter 3

Data Storage

In this chapter, the most common ways how data can be stored by cross-platform interactive 3D content applications are pointed out and briefly described. The final evaluation of these tools and a decision which one I was going to use, will be made in chapter 5.

3.1 Engine Storage

The interactive 3D content applications usually have its own data storage. For instance, *Unity 3D* game engine has special class called *PlayerPrefs* that allows to store and access preferences between game sessions. This way could be sufficient for simple, not complex data storage, as it can set or retrieve only values (integer, float or string data type). The *PlayerPrefs* storage depends on a platform, for instance on *Windows PlayerPrefs* are stored in the special registry, on *WebPlayer* they are stored in binary files. [31]

3.2 Relational Database Management Systems

Relational Database Management System (RDBMS) is a database management system software used for managing the storage, organization, access, integrity of data and security. This system is based on the relation model introduced by E. F. Codd. Relation databases are the most widely used management systems in the world and a lot of currently popular databases are based on this model. Thanks to this system applications can focus on user interface, screen navigation and data validation, while for adding, deleting or moving data the RDBMS is called in time of need. [26]

Object-Oriented Database Management System (OODBMS) is a DBMS storing data logically, based on OO programming language techniques. The main advantage of this approach is the easier integration between the DBMS and the OO languages. In spite of considerably better performance, development and flexibility cost of OODBMS than RDBMS and ORDBMS, it came across to certain serious problems (such as less productivity than with using declarative languages). Object-Relational Database Management System (ORDBMS) uses the features of RDBMS and is taking advantage of OODBMS ideas at the same time. [9]

I have chosen three different RDBMS which I am going to describe further. All of them are using special programming language Structured Query Language (SQL).

3.2.1 MySQL

MySQL is the second most widely used open-source RDBMS and the most popular one of the large-scale database systems (over 100 million copies downloaded or distributed throughout its history). It is well known as a key part of LAMP (Linux, Apache, MySQL, Perl/PHP/Python) - software bundle which is entirely free open-source enterprise software stack, widely used for building dynamic web sites. There are also several paid versions with extended functionality suitable for commercial projects. *MySQL* is used by very popular websites such as *Youtube*, *Wikipedia*, *Facebook*, ... [20] [11]

As RDBMS, *MySQL* allows to store data of many defined data types in two-dimensional data structures called tables. The significant advantage of this relational model and disadvantage at the same time is conceptual simplicity which resulted in intuitive **simple usage**. From the same reason it has some functional limitations and therefore it is not suitable for complex database designs. *MySQL* can be however very **fast** tool, because it does not try to implement the full SQL standard. [33] [20]

3.2.2 PostgreSQL

PostgreSQL is free and open-source advanced object-relational database management system (ORDBMS) well-known for its extensibility and standards-compliance. Even though it is not so widely spread as MySQL, it has experienced community and has won several awards such as The Linux Journal Editor's Choice Award for best DMS or the Linux New Media Award for Best Database System. [27]

As it is not only RDBMS but object-relational DBMS, it provides many of enhancements. For instance, it supports table inheritance, multiple values in a column and calling special functions by SQL statements. Very important attributes of *PostgreSQL* are data integrity, reliability and correctness. It ensures reliability of transactions, as it is fully ACID (Atomicity, Consistency, Isolation, Durability) compliant. *PostgreSQL* provides very powerful features such as Multi-Version Concurrency Control (MVCC), point in time recovery, nested transactions (savepoints), triggers, foreign keys ... [33] [27]

It is higly flexible, as it allows stored procedures or triggers to be load into the database as a library. For developers, there is also a possibility to create their own data types along with functions and operators defining their behavior. Naturally, the *PostgreSQL*'s learning curve for new users can be steeper than of *MySQL*, but for experienced or professional users can be more satisfying. And there are also many tools or libraries created by developers to make the usage of this system more simple. [27]

3.2.3 SQLite

SQLite is embedded file-based RDBMS included in C programming library. It is the world's most widely used RDMS (October 2013), as it is deployed with every *iPhone* and *Android* device and used by the widely spread browsers (*Firefox*, *Chrome*) for meta-data storage. It is included in many of embedded operating systems such as *Microsoft's Windows Phone 8*, *Blackberry's BlackBerry 10 OS*, *FreeBSD*, *Symbian OS* and others, but also in desktop operating system *Mac OS X*. The source code is in the public domain, therefore it is possible to use it for any purpose - private or commercial.

SQLite is SQL database engine using a single local file to store the database and it provides the access to the file by using standard SQL commands. Unlike most other SQL

database management systems, it is not a separate process accessed from the client application, but a fundamental part of it. Due to this fact it is extremely fast and efficient. The *SQLite* database is limited in size to 140 terabytes which is something more than the maximum size limit of a single disk file in many of filesystems. The database file format is **cross-platform**. [11]

SQLite is small (the library can be less than 500KiB with all features enabled), fast and reliable but the primary design goal was the **simplicity** - the simplicity of administration, implementation and maintenance. From that reason it lacks of plenty of useful complex features that enterprise database engines provide such as high concurrency, fine-grained access control, a rich set of built-in functions, stored procedures, esoteric SQL language features, XML or Java extensions, tera- or peta-byte scalability, etc. [11]

The library is also well-known because of its good **reliability**, as it puts emphasis on careful testing and verification. Transactions should be ACID even in case of interruption by power failure or system crash which is verified by special automated tests simulating system failures. *SQLite* uses weakly and dynamically typed SQL syntax not guarantying the domain integrity.

As an embedded database it is stored locally - on individual device and usually as a single file. This approach is suitable when working with **data of single user on one device**. It is ideal for the majority of mobile applications and local games. Using local databases on platforms like WebPlayer can be, however, difficult, because the access to the file system is restricted. The most common solution is to read or download database file into memory. Naturally, the usage of local database can be also inappropriate when user should be able to change the same database from several platforms. In that case, data would have to be updated on all platforms. Even more inappropriate would be trying to use *SQLite* with multi-user applications.

3.3 Serialization

All the mentioned ways of storing data are definitely suitable for dynamic frequently changing data. Although in case of static data or rarely changing data, more appropriate way to store them is to use one of serialization techniques. This approach is common for configuration settings, but also for large-scale static data.

3.3.1 XML

Extensible Markup Language (*XML*) is markup language derived from more complicated *Standard Generalized Markup Language* (*SGML*) which enables creating documents and document systems. *XML* was and still is widely used as an interchangeable data serialization format which offers besides other things these features:

- *XML* is text based and position-independent.
- **Simplicity**: Thanks to its simple and unambiguously structured syntax, it is more readable to humans as well as to machines.
- **Extensibility**: XML enables storing and organizing any kind of information. This is ideal for documents which require to manage various elements of formatting, such as images, graphs, charts, ...

- **Interoperability**: Thanks to its consistent document structures, it is cross-platform. It uses Unicode character set, therefore it supports plenty of writing systems and symbols (runic characters, Chinese ideographs, ...).
- **Openness**: The XML standard is public, completely open, freely available on the web.
- **Error Checking**: Besides necessary meeting minimum syntax requirements, it also offers many useful ways to check the quality of document (such as internal link checking, comparison to document models, ...).

XML also offers a lot of tools for selective extraction, transformation and analysis of data. One of these powerful tools is a query language called *XML Path Language* (***XPath***) used primarily for selecting nodes from an XML document.

XML came across a vast amount of criticism for its complexity and verbosity and in spite of the fact it is still widely used and supported, it is slowly being replaced by alternative text formats such as JSON or YAML. [22] [25] [13]

3.3.2 JSON

JavaScript Object Notation (*JSON*) is one of the most popular *XML* alternatives. It is a data-interchange format based on a subset of *The Javascript Programming Language*. Unlike *XML* it does not include Data Type Description (DTD), but it is designed for data exchange only. Lets swiftly point out comparison with some of important *XML* features:

- **Simplicity**: Data structure of *JSON* data is more standardized, as it has much smaller grammar than *XML*. Whereas *XML* stores data in trees, *JSON* data is stored in records and arrays. It is familiar to objected-oriented languages, therefore the import into these languages is much easier.
- **Extensibility**: *JSON* is not extensible, it is limited to store only classical data such as numbers or text.
- **Interoperability** and **Openness** of both formats are practically at the same level.

In spite of the fact that JSON is without any doubt more suitable tool for data exchange, the majority of web services provides XML encoded data only. Of course, this can be easily solved by conversion of XML data to JSON format and the support of JSON is increasing in time. [13] [15]

Chapter 4

Animations

In this chapter, firstly, a brief description of basic theory of 3D animations related to this project was made. Next, possible solutions how to create animation clips necessarily needed for showing the proper performance of exercises are described.

4.1 Basic Theory

As the animations are essential part of this project, I want to explain here several terms concerning Animations and Animation system in game engines. [30]

- **Animation clip** is data that can be used for animated characters or simple animations.
- **Rigging** is the process of creating the joint hierarchy. Constraints and relationships between objects are made to create controls to manipulate with digital character.
- **Skinning** is the process of connecting a skeleton to the mesh.
- **Retargeting** is applying set of animations created for one model to another.
- **T-pose** is the pose in which the character has his arms straight out to the sides, forming a letter „T“.

4.2 Creating the animation clips

All of the core 3D game engines described in chapter 2 enables to use and manipulate with animation clips. There are many ways how to create the actual clips starting with slow manual transformations of the model and ending with rapid advanced techniques using motion capture. I wrote down the possibilities and made a final evaluation in chapter 5.

4.2.1 Game Engine Tools

All of the mentioned engines (*Unity*, *UDK*, *UE4*) themselves allow to create animations by manual transformations of the model in time line. In spite of not really extensive functionality concerning the animation creation, this approach ensures to avoid problems related to exporting from extern software. Thus, it could mean possibly rapid development.

[1] Source: http://cgcookie.com/blender/2011/12/12/blender-introduction-to-character-rigging/

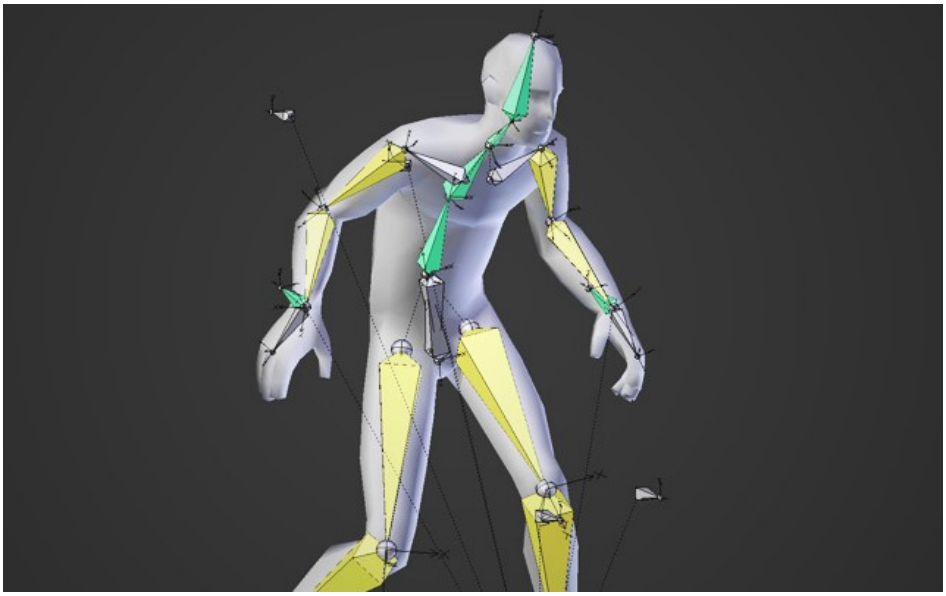


Figure 4.1: Character Rigging in Blender[1]

4.2.2 3D Animation Software

Again, there are many studios suitable for creating 3D animation clips. I have chosen two of the most used and most favorite ones.

a) Autodesk Maya
Maya is a professional award-winning software which provides 3D animation, modeling, rendering, simulation and is widely used in film, architecture and game development. One of the biggest powers of Maya is the animation support. It has plenty of features ranging from advanced rigging (partially automatic) and retargeting to various professional effects. Needless to say that the tool reaching such a high level is expensive. *Autodesk* company, however, has a special offer for students and teachers - full educational version for non commercial use. [1]

b) Blender
Blender is also very powerful 3D modeling and animation software and as well provides a lot of modeling, animation and rendering tools. Without any doubt, the main advantage of Blender is that it is completely free to use for any purpose. It has also a lot of advanced features (such as lightning-fast implementation of UV unwrapping) which are not the part of its commercial counterparts. In its current state, *Blender* can be used for production of high-quality animation. [16]

4.2.3 Motion Capture

Motion capture techniques are frequently used to study clinical problems and musculoskeletal biomechanics, but also for the entertainment industry to create realistic animations. [3]

There are two basic approaches how to use the motion capture technique to get the animation clips:

a) **Marker-less** motion capture approach is designed to allow performance recording without necessary optical scene modifications and to overcome certain restrictions of marker-based approach. In spite of more flexibility brought with marker-less approach, it is difficult to achieve the same level of accuracy as the marker-based methods have. [4]

b) **Marker-based** motion capture approach can provide very accurate results of capturing human performances. It, however, usually requires human actors to wear special clothing, a lot of time time spent on setting up and manual corrections of obtained data. [4]

Figure 4.2: Marker-based Motion Capture (The Lord of the Rings movie)[2]

Although it is still sorely difficult to capture human performances - to map the geometry and motion of actors in the real world onto virtual doubles, it should be enough to make reasonably realistic character animation for this particular project. The marker-less approach is naturally more accessible to regular person, as it only requires a proper motion sensing device and proper software. From this reason, further, I am going to focus on the marker-less approach.

The **Markerless Motion Capture** could be the fastest way, how to create animation clips. Furthermore, there are advanced tools able to capture human motion available. There is one so widely used, I have to mention: ***OpenNI***. This framework is open source software development kit designed for the development of 3D sensing applications which access as it is called *Natural Interaction Devices*. These *Natural Interaction Devices* or *Natural Interfaces* capture body movements or sounds in order to make an interaction of users with computers more natural. [19]

Of course to learn how to work with *OpenNI* and produce accurate reality-like 3D animations can be a long way run.

Until the beginning of 2012, only companies capable to invest vast amount of money for cameras and software, could even consider the motion capture way of creating 3D animations. However, within time, motion capture technique became much more affordable and accessible, as it began to be used significantly more frequently and evolved rapidly. It resulted in releasing affordable and yet very capable software called ***iPI Mocap Studio***. [10]

[2]Actor: Andy Serkis, Source: http://www.serkis.com/performance-capture-gollum.htm

iPI Mocap Studio is a markerless motion capture software tool that uses *OpenNI* middleware and enables to track human body motions and produce 3D animation. Besides the obvious advantages related to markerless motion capture approach (such as no need for special sensor suits) this studio offers these features:

- **Low requirements**: It supports *OpenNI* compliant 3D sensors (*Microsoft Kinect*, *ASUS Xtion Live*, or *PrimeSense Carmine 1.08*) or *Sony Playstation Eye* cameras. It runs on regular PC with gaming class videocard.
- **Easy exporting**: It enables to export 3D animation to most popular formats such as FBX or BVH. It is compatible with the most popular 3D content software such as *Autodesk 3ds Max*, ***Maya***, ***Unreal Engine***, ***Unity***, CINEMA 4D, Lightwave, ***Blender***, ...
- **Portability**: The set-up and calibration of the system can take less than 30 minutes.
- **Tools**: It has integrated motion transfer and animation clean-up tools.
- **Affordability**: Basic edition which includes high quality tracking of the most complex movements (rotations such as dancing, sport, fighting) costs less than 600$.
- and others (not directly related to this project work)

[19] [12]

This tool is highly intuitive and there is no need for programming or spending a lot of time on learning how to use it.

Chapter 5

Evaluation of State of the Art and Design Goals

My primary task is to make a cross-platform interactive 3D application whose purpose is to manifest how Information Technology can help physiotherapists and their patients. Before doing so, I have to select suitable software and approach. This decision should be based on my research described in previous chapters.

In this chapter, I want to sum up the mentioned tools and ways how to create the application. Here I am going to make a decision which tools I am going to use and why I am going to use them. But before I start doing so, firstly it is necessary to briefly describe what I am actually going to do.

5.1 Prototype Design

Usually during rehabilitation sessions, proper exercises are shown to a patient to improve its state and eventually solve its health issue. The problem is that the patients tend to forget the proper performance of exercises or even entire exercises. The main idea is to create a software that shows the patient exercises in correct form as 3D animations. Each patient should obtain its own version of application with set of exercises prescribed by his physiotherapist.

The prototype application should be capable of showing various exercises mutual for all patients (users). This should be possible without Internet connection. Each patient, however, can download sets of exercises prescribed to him by physiotherapist from on-line database. The idea is that the user will get login data and the physiotherapist adds the sets right after each real rehabilitation session. The patient can therefore update the exercise set data stored in his or her device by logging in and after that to use the application in offline mode.

5.2 Core Application

All of the mentioned 3D content software development studios had cross-platform usability, wide support of animations and various 3D model manipulation possibilities. *Unity 3D*, however, has one significant advantage - it can be used for free even for commercial use and also there is no need to pay tax from earnings. That was the main reason why I decided to choose this engine, but not the only one. The greatest power of this engine is definitely the

cross-platform deployment, this free edition supports more platforms than *UDK*. Also the learning curve is easier, as Unity free version toolset is not so evolved as the *Unreal* engine one. In spite of the fact that the free license is significantly limited, I came to a conclusion that it is sufficient for this project work.

As for the *Unreal Engine 4*, without any doubt this engine looks like the most powerful of the mentioned software. It is, however, completely new release and I think it is not ready to be used for my project work yet, especially because of the cross-platform which is still „under construction". I believe that in time it could be the best choice, but I had to choose one engine which was ready right now. Also, the tax from earnings could be a problem.

Beside the cross-platform limitation, unlike *Unity*, *UDK* is also limited by choice of programming language. Unity 3D allows to choose from three programming languages UnityScript, C# or Boo. It is also possible to use all of them, make some scripts in C#, and the others in Boo or UnityScript.

I prefer using one of the languages to avoid possible mistakes and to make the entire project work more readable and editable in future. Therefore I had to choose one language. Even though UnityScript is used much more frequently and unlike the others it appears at all of the official tutorials and documentations, I decided to use C# instead. The main reasons of doing so were two - the expressivity plus clarity of the code and the well specification, standardization of the language itself with quality documentation. I did not want to use Boo, because it appears only at the minimum of tutorials and articles (2014).

5.3 Data Storage

For data mutual for all users - the exercises - I firstly wanted to use local database solution *SQLite* to be able to use SQL syntax and advanced select queries. Although during my research I found out that *Unity 3D* free does not allow to export project to mobile devices while using certain system functions (`System.Data`) crucial for *SQLite*'s functionality. I also came to a realization that serialization approach could be sufficient, as the exercises are actually unchanging static data. Therefore I decided to use *XML* as the main storage of data. I am fully aware of the advantages of *JSON* data format, but I wanted to take advantage of *XPath* for data selection. Also I find *XML* more readable for human and foremost more intuitive. The last thing which led to my final decision was the fact that *XML* is integrated within C# language.

For the login process - downloading exercise set data, it was necessary to use global (on-line) database to store user data. I decided to choose ***MySQL*** because of its simplicity and speed. Even though *MySQL* is not so advanced as *PostgreSQL* and it does not have so many features, it is more than enough for my purpose.

To store loaded user data, exercise sets and other possible secondary data, I decided to use integrated game engine data storage*PlayerPrefs*, as it is the most natural way how to store and access simple data.

5.4 Animations

Using professional 3D animation software can be for sure the best and the most accurate way how to produce high-quality animation clips, as this software is actually designed for this purpose. Unfortunately the price for this software is too high. In spite of the fact that it is possible to use this professional tool for free (as a student), for not large-scale companies

it would be almost impossible to use for commercial product. Therefore, I decided to try free software product - ***Blender*** which also offers promising results in form of high-quality animations.

The *OAMK* provided me *Kinect* depth-sensor device to record data for motion capture. I was convinced that this approach could mean rapid development and could produce above-average quality animation clips. By using this motion capture devices I wanted to obtain an elemental form of animation clips and post-process them afterward. Naturally, the automatic creation of animated clips could be much faster than the manual one, especially when creating a big amount of them.

I also saw using integrated game engine animation tools as potentially fast option. All of the mentioned ways how to create animation clips offers various advantages. As I did not have a real experience with none of these tools, I decided to try all of these three approaches and compare thr results. The details of this experiment and the results are shown in chapter 6.

5.5 Summary

To conclude the decisions made above, as the main software I decided to use free version of *Unity 3D* engine. I have chosen *XML* mark-up language as the main tool for stage of static data and for the exercise sets a combination of integrated engine data storage *PlayerPrefs* and on-line database *MySQL*. As for the production of the animation clips, the decision is yet to be made, depending on the results of experiment described in detail in the following chapter.

I also had to decide which platforms my prototype application will support specifically and what basic functionality it should have.

5.5.1 Required parameters

- Basic functionality:
  - user can interactively choose a muscle group
  - list of exercises attached to selected area is shown
  - 3D animation of chosen exercise plays
  - user can download a set of exercises assigned to him
  - user can choose a model of instructor who is performing the exercises
- Main development environment: Unity 3D game engine
- Programming language: C#
- Data storage: XML, MySQL, integrated game engine data storage
- Supported platforms(deployment): Windows, Linux, Android OS, web browsers (Windows or Mac OS X users only)

Chapter 6

Design

In this chapter I would like to discuss what kind of data I am going to use and how I am going to use them. The second part of this chapter should consist of description how the actual prototype application should look and work according to my design.

6.1 Data Storage

6.1.1 Static Content

As mentioned above, exercises will be stored in XML data format. The design of logical connection between exercises and muscles involved is shown at the picture 6.1.

The table „Exercise“ stores the exercises where each exercise has its own name and there is a particular 3D animation attached to it. There can be three types of exercises:

1. stretching exercises
2. strength exercises
3. other (walk, running, etc.)

There can be also one or more instructions which belongs to each exercise and it is stored in chronological order. It is important because of synchronizing animation with the instructions, as will be discussed further in next chapter.

Each exercise is connected to one or usually more muscles. The meaning of this is the possibility to store complex exercises where more than one muscle group is involved.

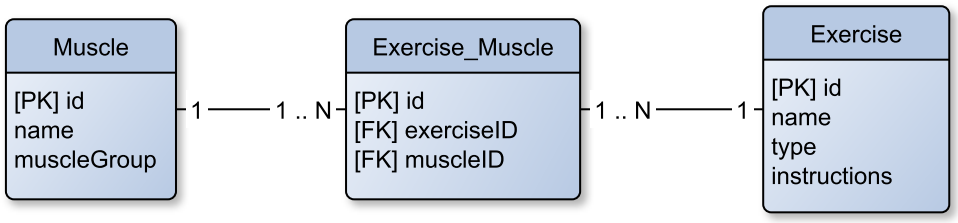


Figure 6.1: ERD for database design

The exercises as well as the instructions were mostly based on literature ([17], [34]) recommended to me by international students of health care and physical therapy.

6.1.2 Dynamic Content

User data - id, login and password are stored on server-side, as well as dynamic exercise sets. Each set includes its id and id numbers of chosen exercises.

6.2 GUI

The OAMK school provided me professional 3D model of human muscular system with detailed high quality textures. I used that model as the first thing user can see and can manipulate with, the usage will be discussed below.

When the application runs, the model of human muscular system and login screen appears. Patient can either log on to the system and by doing that to synchronize the exercise sets stored in his or her device with the server or skip the login process. After that the user can choose a problem area, a muscle group, by clicking (or touching) particular muscle. To make this easy or even possible in case of muscles unseen from the front side, the user can rotate or zoom the camera. After selecting the muscle, a menu box will appear and will be showing possibilities user can choose from as well as the name of the chosen muscle group. By each selecting the muscle group, the certain part of the model is highlighted by green color and the name in the menu is changed.

Basically there are two possibilities how to use the application. The user can either just simply choose Stretching or Strength and the exercises connected to the chosen muscle group will be shown, or he or she can firstly choose the **Exercise Sets** option. In case of the **Exercise Sets** there are shown sets of exercises prescribed by physiotherapist. This approach provides much better usability of the application in real life.

After that, the user can choose one of the exercises (assuming that there are any connected to particular muscle group or exercise set) and the model of human muscular system is hidden and replaced by animated model with skin (instructor). The animation loops and at the same time text instruction steps are shown step by step. The user can pause the animation and play it again at any time.

The user can also select Menu option which will lead him to main menu which offers these options (among others):

- to change model of instructor performing exercises
- to show Exercise Sets

6.3 User Testing

In view of the fact that user testing had a significant impact on the current form of the Graphic User Interface, I decided to briefly describe this topic.

The goal of these tests was not only the usability testing thus to find out how people can use the application while being observed, but also the qualitative research - to get useful opinions or ideas to make the application better.

Based on usability consultant Jakob Nielsen's advice ([18]), I have chosen 5 test users from international students of health care or physical therapy. I created several tasks ranging from those of intuitive character (getting to know with application) to those with specific instructions and goal. I participated these tests as an instructor and as an observer at the same time to ensure the credence of the user testing.

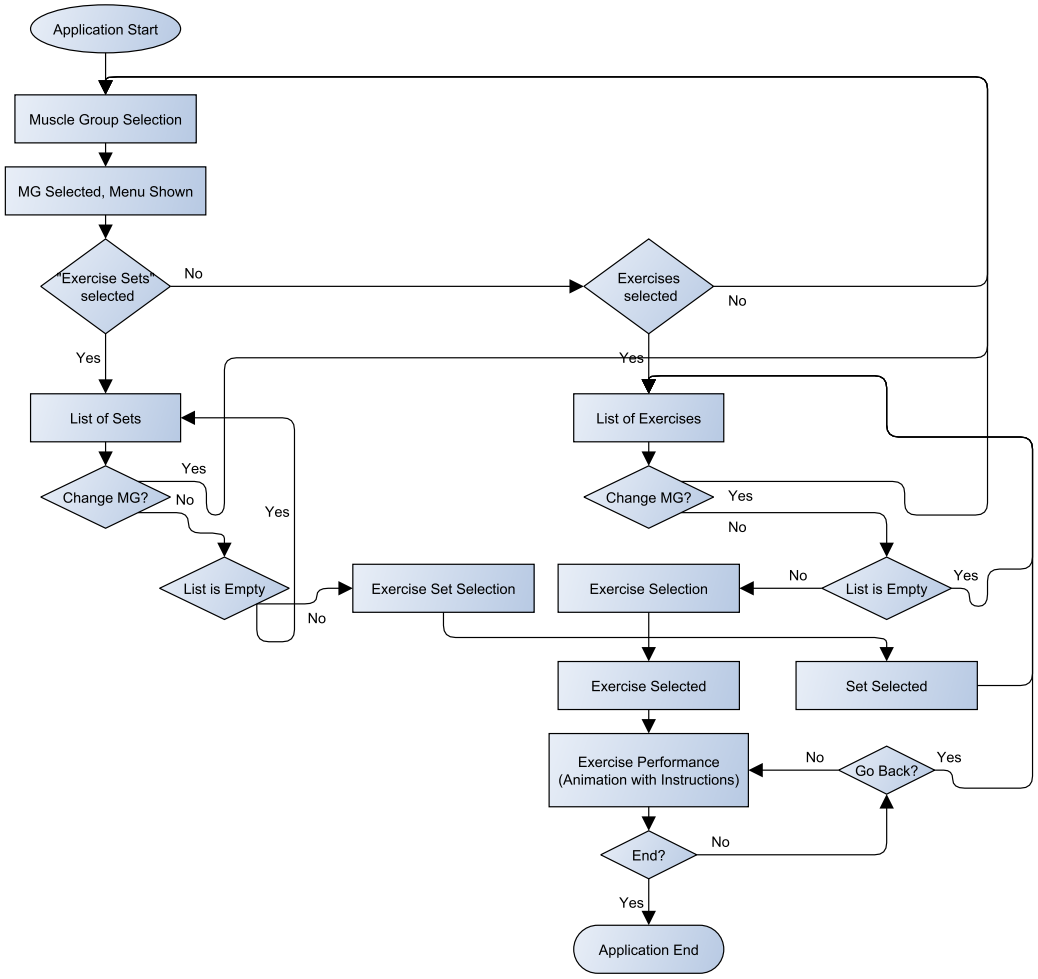


Figure 6.2: Basic user usage of the prototype application

Except for the subjective opinions and notes concerning style and size of font, the tested group provided me really helpful and interesting idea concerning functionality. The first and the best idea was to show the instructions step by step synchronized with the animation, so user could know what and when exactly to perform. The second idea was to enable the user to pause the animation and skip to the next instruction.

Chapter 7

Implementation

Entire application uses only one main scene `MainScene.unity` in Unity 3D project. Main functionality is implemented in `GUIController.cs` script file which is attached to AppController gameobject which is always visible in the main scene.

7.1 Data Storage

7.1.1 Exercises

Static data (exercises) is stored in XML file `data.xml`. I have created my own class `Database` that implements selecting of specific exercises. When the `Database` is initiated, it opens `data.xml` file and afterward it is loaded as an XML document. This class contains among others these functions:

- `SelectExercises` - selects the exercises which belong to proper muscle group
- `SelectByListOfIDs` - selects the exercises by list of identification numbers
- `DBLoadExercises` - calls these two mentioned functions.

These functions are using XPath syntax for selecting matching nodes.

7.1.2 Exercise Sets

After exercises are loaded, `loadUserSettings` function is called and it loads user settings stored in *PlayerPrefs*: `User Name`, `Last ID` - last ID of exercise set stored offline and `Count` - number of exercise sets stored offline.

After user confirms login process by clicking Login button, entered login data along with `lastID` are passed to server side. PHP script `login.php` connects to MySQL database, verify entered username and password and returns three possible result values:

- `done` - There are no more exercise sets available for update.
- `fail` - Error occured.
- else - String of exercise sets available for update.

This result is processed in `login` function. In case of update `parseExSets` function is called which parses the string of exercise sets, updates the `Last ID`, `Count` variables and

saves each set in *PlayerPrefs* called: „Set + `number of set`“ as a string value containing identification numbers of exercises separated by commas.

On server side, there are two tables:

- **user** - consisting of `id`, `username` and `password` (naturally only md5 hash value of password is stored)
- **exercises** - consisting of `id`, `user` (user ID), `exercises` (string of exercise identification numbers separated by commas)

7.2 Animations

My work in this area consisted of these main points:

1) **Obtaining proper 3D models** of human body with muscles, skin and bones
2) **Setting up the humanoid character** for *Mecanim Animation System*
3) **Creating the animation clips** for each exercise
4) **Applying and Controlling the Animations** on the models via animation controller

7.2.1 Obtaining proper 3D models

There is plenty of websites offering variety of 3D models for free (such as [28]). But since I wanted to enable the user a possibility to change appearance of the human model performing the exercises, I had to find a model which meets certain requirements.

I needed 3D *rigged* and *skinned* humanoid type of mesh to use the Mechanim feature Retargeting of Humanoid animations (which means to use the same set of animations from one character model onto another). [30]

Even though I had found a lot of free models marked as fully-rigged humanoid at unofficial websites, the majority of them was unable to collaborate with *Mecanim*. It either did not have enough of bones or did not meet some other requirements (as the animations did not play for some models even when *Mecanim* recognized them as the proper humanoid models). From that reason, I decided to download and use the free **Mecanim-compatible** models from official Unity Asset Store. [29]

7.2.2 Setting up the humanoid character

As mentioned before, thanks to the similarity in bone structure, the *Mecanim Animation System* allows mapping animations from one humanoid skeleton to another and therefore retargeting. To create a retagetable animation, it is however necessary to set up a mapping between the *Mecanim*'s humanoid bone structure which is simplified and the actual bones of the skeleton. This mapping is called an *Avatar*. [30]

The last thing I had to configure was the range of motion of different bones. After setting up the *Avatar* properly, *Mecanim* allowed me to use the *Muscles setup* which is designed for that purpose. The meaning of this configuration is to make an animation look more real and trustworthy, because when it is set in the right way, it can prevent the character from self-overlaps or visual artifacts. [30]

7.2.3 Creating the animation clips

As mentioned earlier, before I created the actual set of animations for the project work, I decided to found out first which tool and which way would show the best results. Therefore, I produced three exercises (simple, medium and complex) with following tools: *Unity 3D*, *Blender* and *Kinect XBOX 360*; and made a comparison of the results as shown at 7.1. It consists of time (in minutes) needed for creating the animation clip with particular tool and also quality rating (1 - worst, 10 - best) of the clip. Ten international Erasmus students were randomly chosen to participate in this research. The „Quality“ is therefore an arithmetic mean value.

Tool		*Unity 3D*	*Blender*	*Kinect XBOX 360*
Clip 1	**Time**	2	3	8
	Quality	10	10	7
Clip 2	**Time**	35	22	8
	Quality	6	8	**10**
Clip 3	**Time**	49	36	8
	Quality	2	6	8
Average	**Time**	29	20	**8**
	Quality	6	10	8

Table 7.1: Results of the research

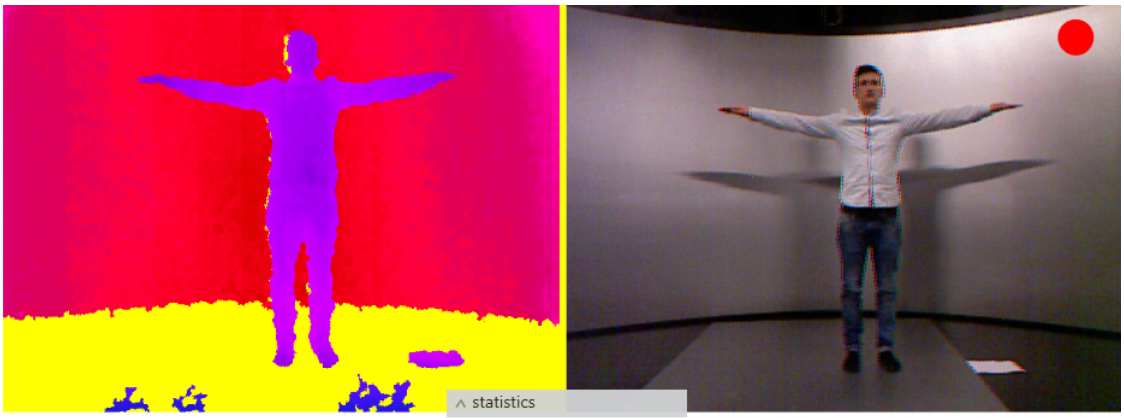


Figure 7.1: Kinect recording, student of physical therapy in T-pose

Based on these results I decided to record the animations with Kinect device. Unity 3D's integrated animation clip production system appeared to be the slowest an the least accurate way how to produce animations. Blender is without any doubt powerful tool for creating animation clips, but Kinect used by *iPI Mocap studio* showed comparable results in significantly better time.

For this recording *OAMK* provided me special studio and one student of physical therapy participated as an actor. The actual process consisted of these phases:

- Setting environment (certain influences could disturb recording such as yellow color or natural light)

- Device calibration + evaluation of background
- Recording of movements (starting with T-pose)
- Setting 3D model to fit with depth record at actor's T-pose
- Tracking starting from T-pose, based on prediction
- Final cut and post-processing (jitter removal)
- Exporting as .FBX file

7.2.4 Applying and Controlling the Animations

The final step was to make each animation clip controllable in *Unity*. I had to create Animation Controller and assign it to each model which should be performing the exercises.

Animator Controller is the main component which enables to set animation behavior by visual state machine. I put there my animation clips that became the states. Between these states can be created transitions with conditions. When the condition is met, the animation clip starts to play. I created my own integer variable `animationNumber` and set condition for each animation clip. This approach enables me to simply change value of this variable (by calling Animator Controller's function `SetInteger` from main script) and make the animation play.

7.3 GUI

For creating a basic Graphic User Interface I used GUI system integrated in Unity 3D, specifically `GUILayout` class and `onGUI` method. Although I had to solve interactive selecting of a muscle group, showing instructions and camera manipulations dependent on target platform.

7.3.1 Selection of Muscle Group

There were several ways how to handle a selection of the muscle group or rather selection of a single one muscle. The easiest but the worst solution at one time, was to attach a script to each sub-model and to use embedded triggers such as `OnMouseDown()` or `OnMouseUp()`, etc. This approach is understandably not ideal, because the model consists of several hundreds of sub-models thus it would meant to create hundreds of files with the same behavior.

The second possible and commonly used solution was to use as it is called ray casting. The raycasting is computer graphics rendering algorithm that uses the geometric algorithm of ray tracing. [23]

In Unity 3D raycasting basically works like forming a line or vector from one point to another in 3D plane. The point is to check if it intersects any colliders or other game objects. This method can provide very useful information such as distances between objects or which game object the camera is looking at. [21]

In view of the fact that the raycasting is supported in Unity, can be used as a function and it is commonly used I decided to choose this solution. I had to assign own Mesh Collider to each muscle sub-model and add tag which contains the muscle group identification. In the end, all the sub-models with the same tag are highlighted, so the muscle group is chosen.

7.3.2 Instructions

There can be several instructions attached to each animation. It was necessary to show the instructions one after another synchronized with the animation loop.
Two options how to solve this issue came to my mind:

a) To make **a loop** that checks the time and is showing the instructions while the animation is playing

b) To use triggers, **animation events** inside particular animation

The first solution would cover all the animations by using the time of instruction loaded from the database. However, this would mean serious performance issue (possibly infinite loop with another loop inside) and probably also problems connected to synchronization. Therefore I decided to choose the second solution.

From the version 4.3, Unity 3D enables a possibility of creating events in animations used in animation controller. It basically means that I can choose certain time in an animation and call certain function. It is, however, limited by calling functions which are part of the actual game object with animation controller attached to it. Thus, I had to attach to this game objects (all the models which should be animated) a short script with function `functionCall` which calls the procedure `nextInstruction` inside the main script `GUIController.cs`. This procedure changes current number of instruction. According to this number the proper instructions are shown. The price for these lower performance requirements was however a must of setting animation events in each animation separately.

7.3.3 Camera Manipulations

For manipulations with main camera I have created script `Camera.cs` that implements two following functions:

a) **Zooming**

To make zooming functional I basically needed only to adjust a value of Main Camera's variable `fieldOfView`. Naturally I had to set some limitation of this value as well as to make the change dependent on user's interactions.

For standalone platforms and Webplayer I took advantage of integrated triggers reacting to Mouse ScrollWheel.

For mobile devices I had to choose different approach. I used `Input.touchCount` function to indicate two touches of the screen at once. To initiate the zooming, next requirement had to be met: both of these touches had to move which I indicated thanks to `phase` attribute of the `Input.GetTouch` function. After that I calculated difference in current and previous distance of touches to decide whether it should zoom in or zoom out.

b) **Rotation**

To make the camera rotate around the model I had to transform it dynamically by using `RotateAround` function. I set the variables of this function the way to limit it only by horizontal movement. The only changeable variable was an angle of rotation.

For standalone platforms and Weblayer I used `Input.GetMouseButton(0)` trigger to initiate the rotation. I obtained the angle degrees from `Input.GetAxis` function which returns the value of the virtual axis.

For mobile devices I used `Input.touchCount` function and `Input.GetTouch(0).phase` to initiate the rotation after touchpad movement was made. I calculated the angle from the delta position of the starting point and ending point by using `deltaPosition` attribute of `Input.GetTouch` function.

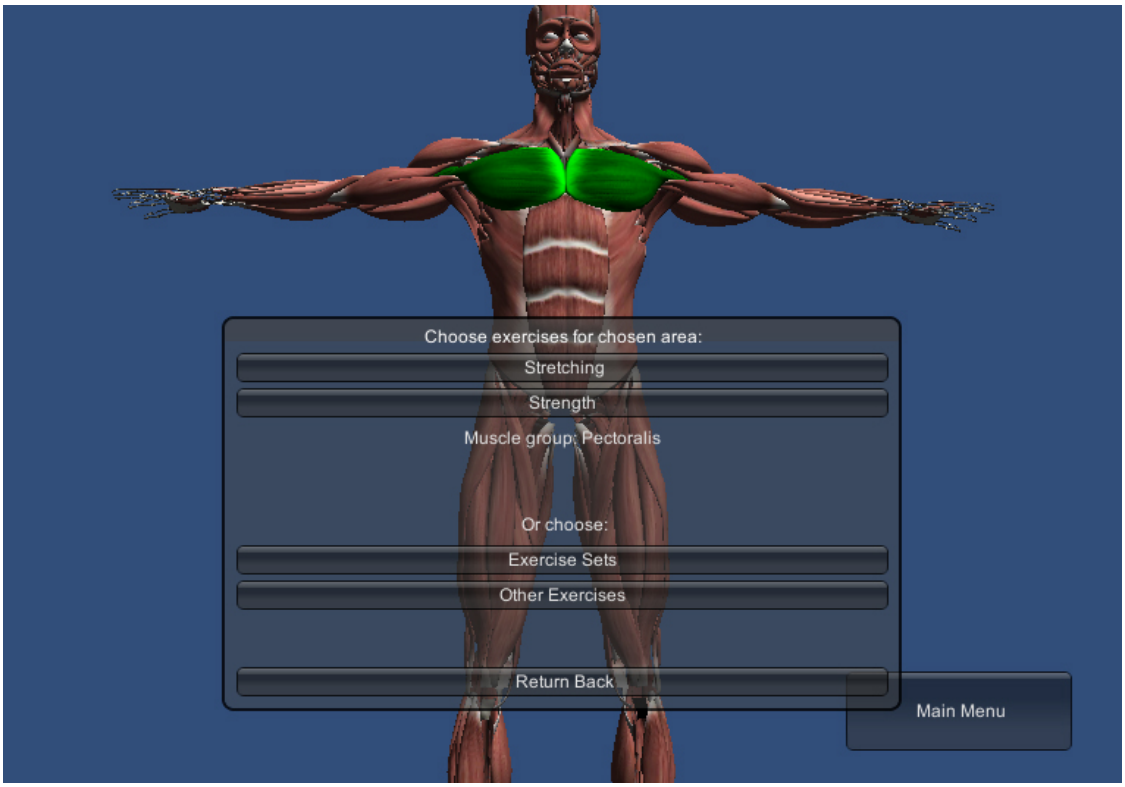


Figure 7.2: The Prototype Application GUI

7.4 Evaluation of My Work

Platform	**Static Data Size**	**Executable File Size**
Android	<30MB (.apk)	
Linux, 64/32bit	<50MB	<20MB
Windows, 64/32bit	<50MB	<15MB
Unity Web Player	<30MB	

Table 7.2: Prototype Metrics

7.4.1 Recording

By using *iPI Mocap Studio*, we managed to make more than 80 recordings, but only 33 of them were usable for animation production. We encountered a problem that made impossible to use certain exercises. When actor's arm or leg was not visible for even a moment, iPI Mocap's tracking based on prediction failed, as it lost track of this part of the body. It resulted in critically deformed animation. It took us a lot of time to figure out, from what angle it is most efficient to record. Of course, by using several of depth-sensor devices, the result could be significantly better.

7.4.2 Exercise Import

The importing of new exercises and animations is not enabled from certain reason: The entire concept of the application should be for free, but potential rehabilitation company interested in this product can buy set of exercises made to measure. For instance, physiotherapist could be invited to a studio to show the movements and make sure everything was recorded correctly. Or the interested company can provide videos of exercises and just let the developers to handle the rest. Naturally, there are many possibilities how to sell the final product based on the main idea of the prototype.

7.4.3 General

The prototype application is capable to manipulate with 3D model of human muscular system, updating and showing the exercise sets prescribed by physiotherapist, showing in total more than 30 exercises along with animations and changing instructor model. It is runnable on *Linux*, *Windows*, *Android* and also on web browsers that supports *Unity Web Player*. It was tested on these mentioned platforms, but could be also easily exported to another ones such as *BlackBerry* or *Windows Phone 8*. Static data (exercises) is stored in internal memory of devices.

Chapter 8

Conclusion

The main goal of this work was to create a prototype of cross-platform 3D interactive application working with 3D models and animations. This system simulates a rehabilitation session and demonstrates 3D visualization possibilities in field of physical therapy.

I managed to create such a system along with more than 30 relatively realistic looking 3D animation clips.

After I got acquainted with the most powerful tools and ways how to create such a cross-platform 3D interactive application, I decided to use the free version of *Unity 3D* game engine as a core software. It allowed me to make the application runnable on many platforms, the limitations of this free license, however, forced me to use *XML* serialization for storing exercise data instead of local database which would be probably more appropriate for advanced exercise selection. Entire animation production based on marker-less motion capture took more time than I expected, and since I used only one depth-sensor camera, it resulted in certain limitations and impossibility to track many of moves. The prototype application was tested on standalone devices, mobile devices and also on certain web browsers.

As for the future, I can imagine much wider usage of depth sensor devices. The potential customer in form of rehabilitation company can borrow or sell this device to their patients and offer them more possibilities. For instance, the patient can use this device to track his or her moves while the application instructor is showing the correct performance. The user could not only see himself and make his own comparison based on what he or she sees, but the application could also offer biomechanic analysis of his or her movements. This could result in improvement of patient's performance, as well as in its self-confidence, but also in motivation and mood to actually do the exercises. This concept of „game“ can go naturally even further or this analysis could be recorded and serve physiotherapist as the evidence of patient's effort and correct performance.

Bibliography

[1] Autodesk Inc. Autodesk website. `http://www.autodesk.com/`.

[2] Sue Blackman. *Beginning 3D Game Development with Unity 4*. Apress, second edition, 2013. ISBN 978-1-4302-4899-6.

[3] S. Corazza, L. Mundermann, A. M. Chaudhari, T. Demattio, C. Cobelli, and T. P. Andriacchi. A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach. *Annals of Biomedical Engineering*, 34:1019–1029, 2006.

[4] Edilson De Aguiar, Carsten Stoll, Christian Theobalt, Hans-Peter Seidel, and Sebastian Thrun. Systems, methods and devices for motion capture using video imaging, February 26 2013. US Patent 8,384,714.

[5] DevMaster. Unreal development kit (udk), 2012.

[6] John P. Doran. *Getting Started with UDK*. Packt Publishing, 2013.

[7] Epic Games, Inc. Unreal development network. `http://udn.epicgames.com`.

[8] Epic Games, Inc. Unreal engine website. `http://www.unrealengine.com/`.

[9] Johannes Gehrke and Raghu Ramakrishnan. Database management systems, 2003.

[10] Joe Herman. A review of ipi soft's markerless motion capture system, 2012.

[11] Hipp Wyrick & Company Inc. Sqlite official website. `http://www.sqlite.org`.

[12] iPi Soft LLC. ipi soft website. `http://ipisoft.com/`.

[13] JSON.org. Official json website. `http://www.json.org/`, 1998.

[14] Aaron Lee. 15 essential mobile game development tools. *Develop*, 10:21–25, 2013.

[15] Kasia Mikoluk. Json vs xml: How json is superior to xml, 2013.

[16] Tony Mullen. *Introducing character animation with Blender*. John Wiley & Sons, 2011.

[17] Arnold Nelson and Jouko Kokkonen. *Stretching anatomy*. Human kinetics, 2007.

[18] Jakob Nielsen. Why you only need to test with 5 users, 2000.

[19] OpenNI. Openni website. `http://www.openni.org/`.

[20] Oracle Corporation. Mysql official website. http://www.mysql.com/about.

[21] Prof. Fabian Winkler. Computer games course. http://web.ics.purdue.edu/~fwinkler/AD41700_F13.

[22] Erik T Ray. *learning XML*. O'Reilly Media, Inc., 2003.

[23] Scott D Roth. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18:109–144, 1982.

[24] Geoff Sholler. *Build a Game with UDK*. Packt Publishing, 2013.

[25] Simon St.Laurent. Why xml? http://www.simonstl.com/articles/whyxml.htm, 1998.

[26] Sai Sumathi and S Esakkirajan. *Fundamentals of relational database management systems*, volume 47. Springer, 2007.

[27] The PostgreSQL Global Development Group. Postgresql official website. http://www.postgresql.org.

[28] TurboSquid. Turbosquid website. http://www.turbosquid.com/.

[29] Unity Technologies. Unity asset store. http://www.assetstore.unity3d.com.

[30] Unity Technologies. Unity official documentation. http://www.docs.unity3d.com.

[31] Unity Technologies. Unity official website. http://www.unity3d.com.

[32] Dave Voyles. *UnrealScript Game Programming Cookbook*. Packt Publishing, 2013.

[33] Yang Xiaojie. Analysis of dbms: Mysql vs postgresql, 2011.

[34] Mary M. Yoke and Carol Kennedy. *Functional Exercise Progressions*. Coaches Choice, 2003.

Appendix A

CD content

Path	Description
`/source_files/`	Source files: • `/Rehab/` - source files of the prototype application (Unity project data) • `/Server/` - source code files related to server side (just for demonstrative use)
`/executable_files/`	Executable files: • `/Android/` - for Android OS • `/Linux/` - for Linux (64/32) bit OS • `/Web/` - for web browsers • `/Windows/` - for Windows 7 (64/32) bit OS
`/tex/`	Source code files of this document
`/thesis.pdf`	PDF file that contains this document
`/readme.txt`	Text file that contains user and developer manual

Appendix B

Prototype Application Screenshots

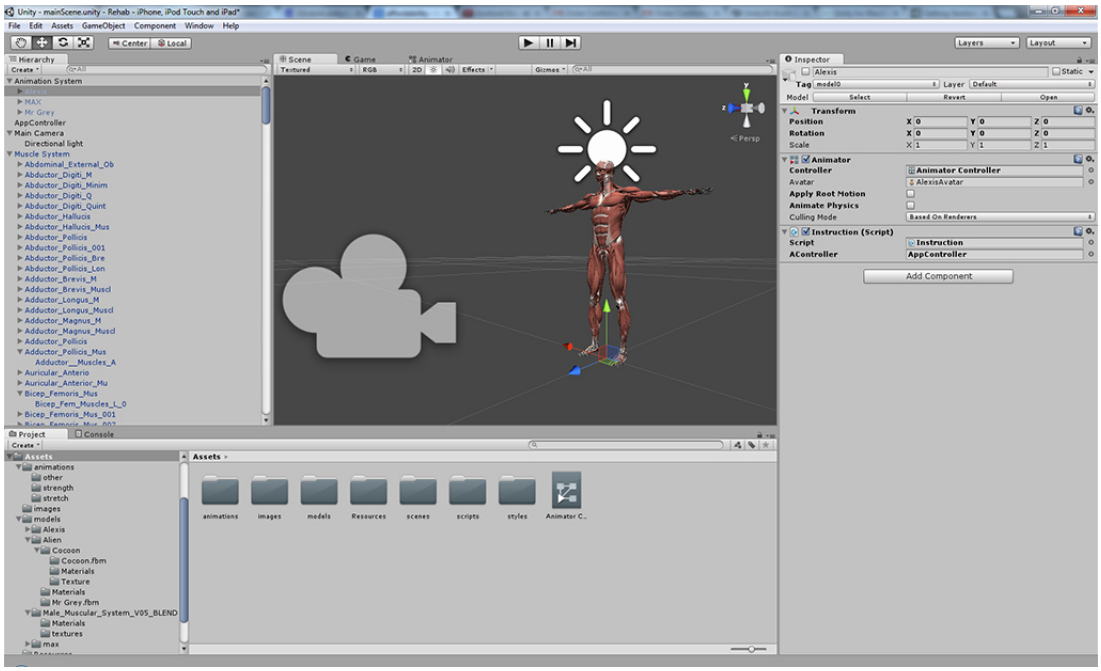


Figure B.1: Prototype Development in Unity 3D - Muscular System

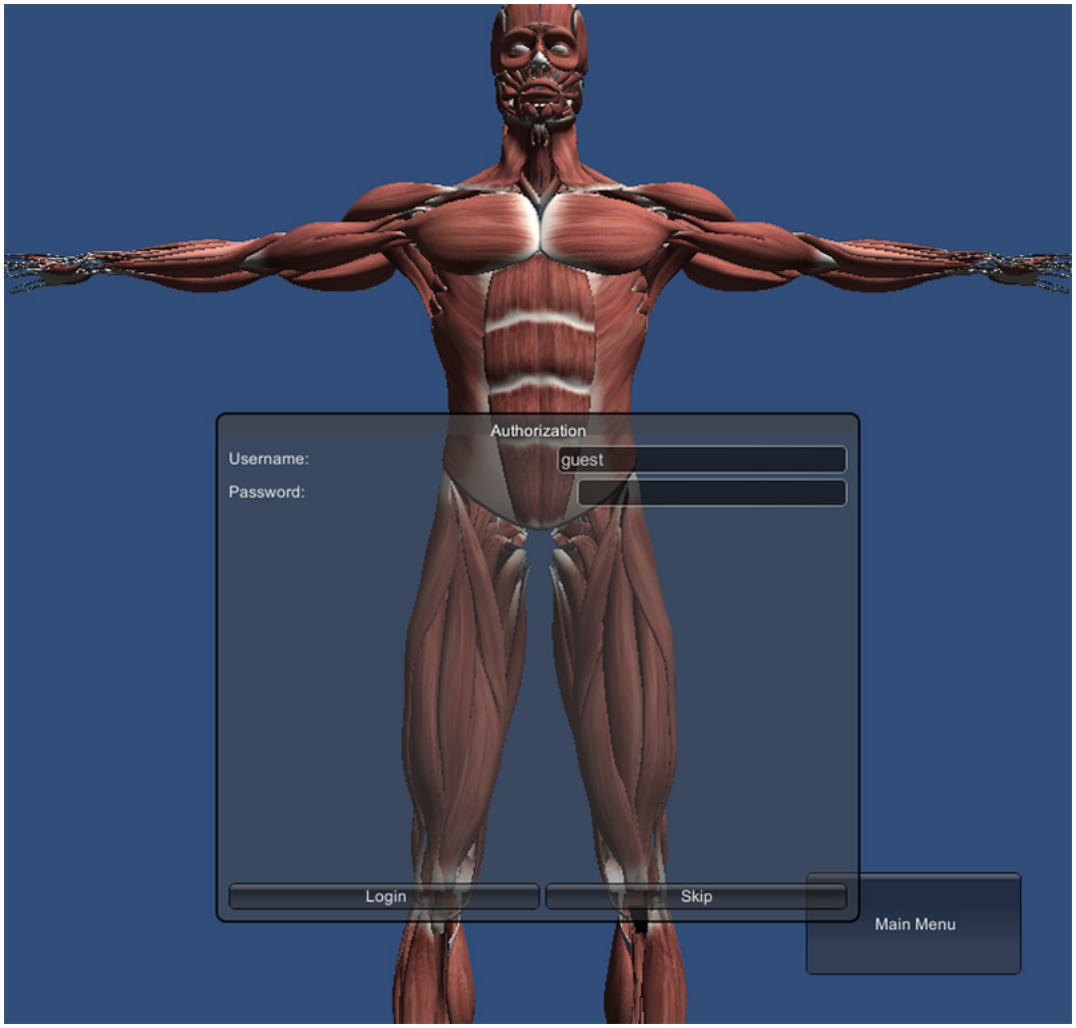


Figure B.2: Prototype Application - Login screen

Figure B.3: Prototype Application - Instructor performing an exercise