TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Department of Mechatronics

Chair of Mechatronics systems

MHK70LT

Maksimilian Tarasevich

VIRTUAL OBJECTS INTEGRATION AND MOTION RECOGNITION FOR AUGMENTED REALITY APPLICATIONS

MSc thesis

The author applies for
the academic degree
Master of Science in Engineering

Tallinn

2016

Author's declaration

I declare that I have written this graduation thesis independently.

These materials have not been submitted for any academic degree.

All the works of other authors used in this thesis have been referenced.

The thesis was completed under Robert Hudjakov and Eduard Petlenkov supervision

Author

_____ signature

The thesis complies with the requirements for graduation theses.

Supervisor

_____ signature

Supervisor

_____ signature

Accepted for defence.

_____ Chairman of the defence committee

_____ signature

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Mehhatroonikainstituut

Mehhatroonikasüsteemide õppetool

MHK70LT

Maksimilian Tarasevich

VIRTUAALSETE OBJEKTIDE INTEGREERIMINE JA LIIKUMISTE TUVASTAMINE TÄIENDREAALSUSEGA SEOTUD RAKENDUSTE JAOKS

MSc Lõputöö

Autor taotleb

tehnikateaduste magistri

akadeemilist kraadi

Tallinn

2016

TUT Department of Mechatronics
Chair of Mechatronics systems

MSc THESIS TASK SHEET

2016 spring semester

Student Maksimilian Tarasevich 143764MAHM

Study program Mechatronics

Specialty -

Supervisor Research Scientist Robert Hudjakov, Associate Professor Eduard Petlenkov

Consultants Research Scientist Aleksei Tepljakov, Associate Professor Kristina Vasiljeva

THESIS TOPIC:

(In Estonian) Virtuaalsete objektide integreerimine ja liikumiste tuvastamine täiendreaalsusega seotud rakenduste jaoks

(In English) Virtual Objects Integration and Motion Recognition for Augmented Reality Applications

Assignments to be completed and schedule for their completion:

No.	Task description	Deadline
1.	Data study, pre-processing and feature selection	01/03/2016
2.	Programme creating and components connecting	01/04/2016
3.	Programme evaluation, testing and deployment	01/05/2016

Engineering and economic problems to be solved:

The problem to be solved is to devise a reasonably successful programme for motion recognition and virtual object integration to augmented reality.

Language of the thesis: English

Deadline for submission of the application for defence: 16/05/2015

Deadline for submission of the thesis: 20/05/2016

Student	Maksimilian Tarasevich	signature:	date: 12/05/2016
Supervisor	Eduard Petlenkov	signature:	date: 12/05/2016
Supervisor	Robert Hudjakov	signature:	date: 12/05/2016

Confidentiality requirements and other corporate terms and conditions shall be set out on the reverse side.

TABLE OF CONTENTS

EESSÕNA **8**

FOREWORD **9**

1. INTRODUCTION **10**

1.1 Background 11

1.2 The problem definition 12

1.3 Methodology 12

1.4 Outline of thesis 12

2. STATE OF ART **14**

2.1 Augmented Reality Chess Game 14

2.2 HoloLens and Citigroup 15

3. INTRODUCTION TO AUGMENTED REALITY APPLICATION **16**

3.1 The Augmented Reality concept 16

3.2 Classification of augmented reality systems 17

3.3 Augmented reality hardware 19

3.3.1 Displays 20

3.3.2 Input devices 21

3.3.3 Tracking devices 22

3.3.4 Computers 22

3.4 Conclusion 22

4. COMPONENTS OF THE SYSTEM **23**

4.1 HMD 23

4.2 Camera 23

4.3 Motion recognition system 25

4.3.1 Overview of notion recognition systems 25

4.3.2 Review of alternatives for motion recognition 26

4.3.3 Conclusion 27
4.4 Personal Computer 27
4.5 Conclusion 27
5. SELECTING FRAMEWORK FOR DEVELOPMENT OF THE APPLICATION ... 29
5.1 Game Engines. Review of alternatives 29
5.1.1 Unity 5 30
5.1.2 CryEngine 3 30
5.1.3 Unreal Engine 4 31
5.1.4 Conclusion 32
5.2 Additional plugins for augmented reality application 33
5.2.1 OpenCV 33
5.2.2 Vuforia 34
5.2.3 ARToolKit 34
5.2.4 Comparison of plugins 35
5.3 Conclusion 35
6. VIRTUAL OBJECT CREATION 37
6.1 Software for virtual object creation 37
6.1.1 3ds MAX 38
6.1.2 Maya 38
6.1.3 Blender 38
6.1.4 Results of comparison 39
6.2 Own object creation 40
6.3 Virtual object integration 42
6.4 Conclusion 44
7. APPLICATION DEVELOPMENT 45
7.1 UE4 and Oculus Rift 45

7.2 Marker method. ARToolKit plugin integration and use 45
7.2.1 Markers for ARToolKit 47
7.2.2 Own marker creation and integration 50
7.3 Markerless method. OpenCV plugin 54
7.4 Plugin integration into UE4 54
7.5 Markerless application concept and level creation in UE4 55
7.5.1 Concept 56
7.5.2 Level creation and Oculus Rift functionality integration 57
7.5.3 Using OpenCV library to display video stream 57
7.5.4 Our own virtual object integration and positioning 60
7.5.5 Application testing and conclusion 61
7.6 Augmented reality application for laboratory work 62
7.7 Conclusion 63
8. MOTION RECOGNITION SYSTEM 65
8.1 Kinect V1 integration 65
8.2 Kinect V2 67
8.3 Conclusion 68
9. STEREO CAMERA INTEGRATION 69
CONCLUSION 71
KOKKUVÕTE 73
REFERENCES 75
APPENDICES 80
Appendix 1. C++ code for web camera integration to UE4 80

EESSÕNA

Käesolev magistritöö on kirjutatud Tallinna Tehnikaülikoolis mehaanikateaduskonnas, mehhatroonikainstituudis koostöös automaatikainstituudiga.

Peamiseks eesmärgiks oli täiendreaalsuse rakenduse loomine, mis toetaks liikumiste tuvastamist. Samuti töö eesmärgiks on virtuaalse objekti rakendusse integreerimine.

Selleks, et saavutada püstitatud eesmärke, peab lahendama järgmisi probleeme:

1. Valida sobivad komponendid ja tarkvara
2. Luua realistlikku objekti
3. Integreerida virtuaalset objekti rakendusse
4. Integreerida rakendusse liikumistuvastussüsteemi
5. Leida kõige sobivamat meetodit täiendreaalsuse rakenduse jaoks
6. Ühendada kõik süsteemi osad omavahel

Käesoleva töö autor tahab tänada oma juhendajaid ja konsultante toetuse ja igakülgse abi eest: professor Eduard Petlenkov, teadlane Robert Hudjakov, teadlane Aleksei Tepljakov, professor Kristina Vassiljeva.

Samuti tahaks tänada mu vanemaid ja naist toetuse, heade sõnade ja kannatlikkuse eest.

FOREWORD

This master thesis is written at the Tallinn Technical University, Faculty of Mechanical Engineering, Department of Mechatronics and in collaboration with Department of Computer Control.

The aim of this thesis is to develop an augmented reality application with motion recognition system, which will transmit a body position to the application. In addition, inside this application we have to integrate a virtual object with realistic texture and set position inside the real world.

To achieve this goal it is needed to solve the following problems:

1. Choice of optimal components(hardware) and software
2. Creation of realistic virtual object
3. Integration of the virtual object into application
4. Integration of motion recognition system into application
5. Find most suitable method for augmented reality application
6. To unite all parts of the system

First and foremost I wish to thank my supervisors, associate professor Eduard Petlenkov, research scientist Robert Hudjakov, research scientist Aleksei Tepljakov, associate professor Kristina Vassiljeva. They have been supportive since the first day I began my master thesis.

I would also like to thank my parents and girlfriend. They were always supporting me and encouraging me with their best wishes.

The thesis is in English and contains 79 pages of text, 9 sections, 18 figures, 5

tables and 1 appendix.

1. INTRODUCTION

Augmented reality is a one of the promising direction in IT industry. Presently this word is perceived as a younger brother of virtual reality, but it is necessary to note at once that they are different and independent fields of art.

This technology is a completely new way to access data and is highly likely to have major impact on society, which is comparable to the effect of the advancement of the Internet. The authorship of the term "augmented reality" belongs to Tomas Preston Caudell, Boing research laboratory engineer; in 1992 he applied the principles of augmented technology in a system, to assist workers in installing electrical cables in aircraft [1].

Augmented reality – is a technique of augmentation – adding and supplementing digital content over the real world using computers and typically mobile audio-video devices, like data glasses. It is a subdivision of mixed reality, which stands for mixing computer generated content – a virtual world – into the real world. Heimo et al. have discussed ethical issues arising from these developing technologies [2]. This technology has a potential to make the relationship of people with information more ergonomic - data can be automatically delivered to users in the correct context for different situations in everyday life, so that the technology will raise the interaction between human action and information on a fundamentally different level.

Augmented reality is no longer substance of science fiction - it is already being implemented in mobile applications for smartphones and tablets. However, the greatest hopes and prospects for augmented reality are associated with creation of a type of hands-free device that has certain advantages for a specific range of users. This technology will be in demand in different areas ranging from armed forces and police, to solve problems better orientation in the dark with the imposition of the image area of the thermal images of people and objects from the navigation parameters prior to surgery, constantly having in view the vital signs of the patient during surgery, as well as the mechanics of performing complex repair operations, and even for nursing mothers, whose hands are busy child, but at the same time required to perform the daily hassles. However, creation of serial hands-free devices for augmented reality is problematic. It is believed that the level of development of modern electronics has not yet reached required level.

The present thesis is devoted to study of creation of augmented reality applications with an integrated motion recognition system. In particular, markered and markerless augmented reality applications methods will be considered. Implementation of different ways of human motion recognition and integration into the augmented reality system are investigated.

In addition, experiments with different methods of augmented reality applications with integrated motion recognition system are conducted and the obtained results are analysed.

1.1 Background

The technology behind augmented reality becomes more and more popular with each passing day and is increasingly used in various fields. This technology has a great potential and this is precisely why it is actively developing.

In augmented reality the user is looking at the real world through the camera, and thus the real image is complemented by virtual objects that are integrated into the physical environment. Opportunities to display virtual objects and interact with them are endless. This type of application can be use in the following areas:

1. Medicine
2. Education
3. Military
4. Applied science
5. Sport
6. Game industry

We can use experience and information from this thesis to create an application for laboratory works. For example we can take laboratory work with magnetic levitation system- this equipment is very expensive and in the author's home university there is only one piece of such equipment. However, by using the application developed throughout this thesis we can provide possibility to replace different expensive equipment on the virtual model of this equipment.

1.2 The problem definition

This thesis is devoted to researching possibilities to create augmented reality application with a motion recognition system. The goal of this thesis is to design an augmented reality application with own realistic virtual object and a motion recognition system.

To achieve this goal the following problems need to be solved:

1. Choice of optimal components and software
2. Creation of realistic virtual object
3. Integration of a virtual object into the application
4. Integration of a motion recognition system into the application
5. Search for the most suitable method of augmented reality application
6. Unification of all parts of the system

The result of the thesis is an augmented reality application with a motion recognition system. We achieve that by doing research and analysis of each step of development of this type of application and by identifying and using the most optimal solutions.

1.3 Methodology

This thesis is based on criteria method and practical approach. Criteria method means that we declare criteria and based on them make a choice. Practical approach, in the context of this work, is an approach in which we pay more attention to developing a real application rather than reviewing extensively underlying theory. Our approach should help us to reach real result and experience.

1.4 Outline of thesis

The thesis is structured in such a way as to clearly provide information about each stage of the application creation and present them in a logical sequence.

In the first chapter we introduced the course of the thesis, determine goals and problems. At the same time we outlined the approach to achieve the objectives.

In the second part will review the most interesting application, in our opinion. A chess game application was done during the university research. The second review is based on the HoloLens application for banking sector.

The introductions into AR application chapter will be devoted to the review of AR concept, AR system classification and AR devices. This chapter will help to determine merits and provide background information about the AR.

The fourth chapter is devoted to review all the required components for the system. In addition, for each component, criteria will be determined and based on the comparison a choice will be made.

In the fifth chapter we will consider different frameworks for AR application creation. We will explore different frameworks and using the criteria method we will chose more suitable frameworks for application.

The virtual object creation chapter is devoted to providing information and practical experience about how to create a suitable realistic virtual object and integrate it into the application.

The seventh chapter is the significant part in the thesis. In this chapter we will create own application using different methods of AR. In the end of the chapter we will compare the methods and will provide pros and cons for each one.

The motion recognition system chapter will consider system for motion capture and how to transmit signal to application for application charter position changing.

The ninth chapter is devoted to providing information about stereo camera integration into the application.

In the last chapter or conclusion, we will summarize all received information and will review the goal achievement.

2. STATE OF ART

This section is devoted to existing research on augmented reality application.

2.1 Augmented Reality Chess Game

In article „*An Interactive Augmented Reality Chess Game using Bare-Hand Pinch Gestures"* [3] an augmented reality chess game with gestures recognition system for interaction with the game was reviewed. The main principle of this game application is same as in a usual chess game. In context of the application the player has to be able to see all virtual chess objects at the same time in order to figure out what the next move is going to be.

The authors used marker tracking method to achieve high precision; In this case they used maker board form ArUco library. As a motion recognition system Real Sense 3D camera was used (they used this name for a stereo camera). As the key gesture pinch gesture was determined, because it is one of more intuitive gestures for interacting with a digital object. Image with the virtual chess and environment was displayed on PC screen and on Oculus Rift. Additional feature was handling occlusion for mixed reality environmental – augmented reality space should be taken into consideration so that virtual content can blend with the natural environmental. Principle of work is shown on the Figure 2.1

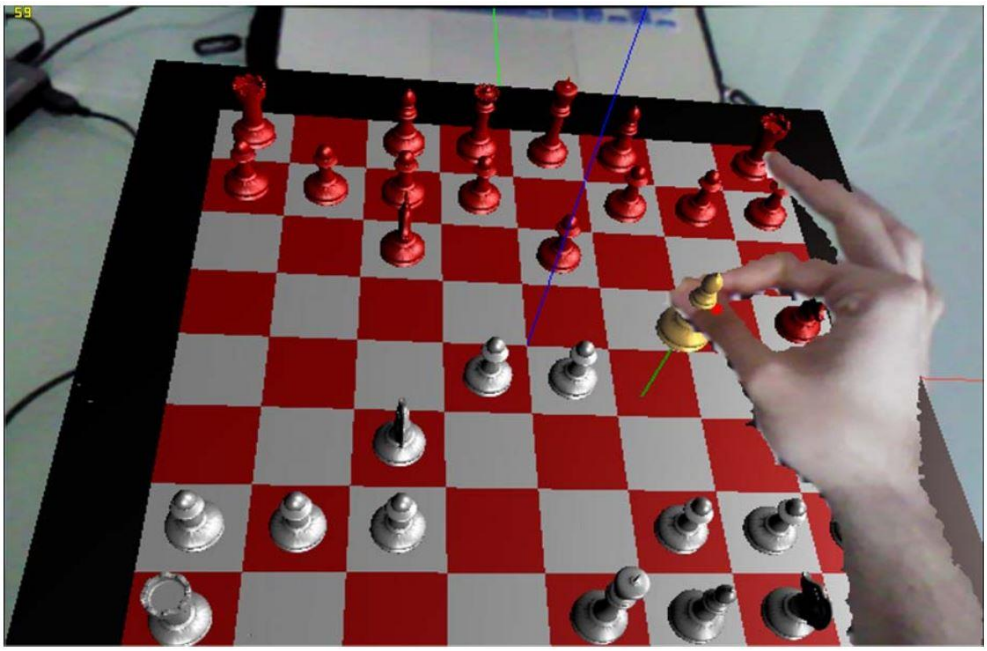

Figure 2.1 Chess game [3]

2.2 HoloLens and Citigroup

In this application the world of augmented reality faces the world of stock trading. The company 8ninths has developed a holographic workstation using HoloLens glasses for Citigroup, one of the largest financial institutions. Holographic station is designed to enhance work efficiency. Applying HoloLens technology allows to display financial data in real time in the form of 3D holograms. This system consists of three levels of information that are dynamically updated in order to cooperate, which in turn allows traders to examine, manipulate, and interact with a large amount of data in the combined 2D and 3D environments. The workstation is connected to a device Citi, company has access to its data and consists of a plate, the space to display 3D holograms, keyboard, mouse gesture recognition devices, speech and sight. [4]

3. INTRODUCTION TO AUGMENTED REALITY APPLICATION

3.1 The Augmented Reality concept

The authorship of the term “augmented reality” belongs to Tomas Preston Caudell, Boing research laboratory engineer. In 1992 he applied the principles of the augmented technology in a system meant to help workers in installation process of electrical cables in aircraft [1].

Figure 3. 1 Mixed reality concept [5]

The essential difference of augmented from the virtual reality is preservation of the physical world as the context in which virtual objects are represented and with which they interact (see Figure 3. 1 Mixed reality concept Figure 3. 1). Virtual reality is completely abstracted from the physical world to put the user completely in the virtual world. Virtual reality uses special positional trackers with displays (virtual reality glasses) that dynamically update user-visible space in the virtual environment.

It is important to understand that augmented reality completely changes this paradigm, and as a result the virtual objects are placed in the real user environment.

Thus, the augmented reality is a technology that complements an image of real objects with various objects of computer graphics, as well as combines images from different sources of computing environment: camera, accelerometer, compass, etc. The environment scheme of augmented reality is presented in the figure below (see Figure 3.2). In contrast to the "virtual reality", which involves completely artificially synthesized world (video), augmented reality

involves the integration of virtual objects in the natural video scenes.

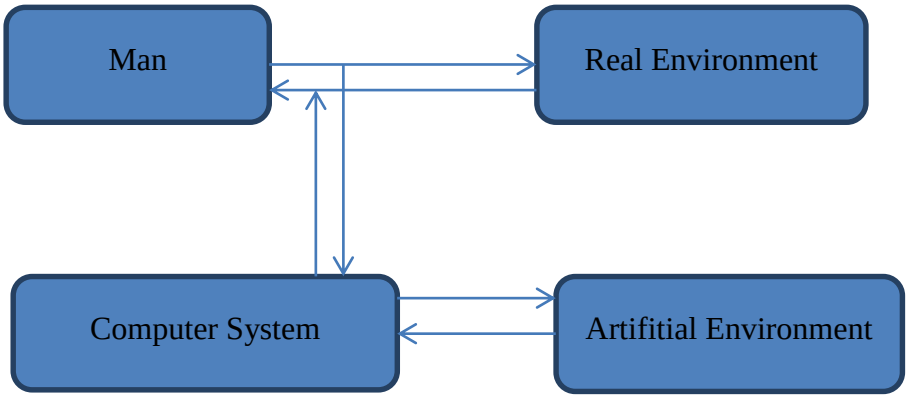

Figure 3.2 Scheme of AR environment

Ronald Azuma identified a number of features that augmented reality should have [6]:

1. A combination of the real and the virtual world;
2. Interactivity;
3. Three-dimensional representation of objects

3.2 Classification of augmented reality systems

A man gets an idea of the surrounding area with a large set of senses. Augmented reality system, being a mediator between a man and reality, should create a signal to one of these senses. Thus, by type of information representation augmented reality systems are:

1. Visual. They are based on human visual perception. The objective of such systems is to create an image that will be used by a man. Because the image for a man is more informative and understandable, this type of system is more common.
2. Audio. These systems focus on auditory perception. Features of such systems are considered in the article "*Development and Evaluation of Augmented Reality Audio Systems"* [7]. Most of these systems are used for navigation. For example, they give special signals when a person reaches a certain place. One can use stereoscopic effect that allows a person to go in the right direction, focusing on the source of the sound. An example of such a system is Hear & There [8].
3. Audio-visual. It is a combination of the two previous types; however, the audio information in them only has auxiliary character.

The Augmented Reality System is always in need of the information received from the real environment - the virtual objects are based on these data. Each of these systems has a specific set of sensors - devices that allow gathering information from the environment: sound and electromagnetic waves, acceleration, etc.

To classify them it makes sense to separate the sensors not on the types of physical quantities, but on their purpose, because similar in nature signals can carry different information. According to the type of sensors there are the following systems:

1. Geo position systems focus primarily on the signals of GPS positioning systems. In addition to the receivers of such signals geo position system may use a compass and accelerometer to determine the rotation angle relative to the vertical and azimuth.

2. Optical systems process the image obtained from the camera, which can be moved with the system or independently of it.

3. Audio systems process the audio input form the microphone.

Augmented reality can be distinguished by degree of interaction with the user. In some systems the user plays a passive role; he oversees a system response to changes in the environment. Other systems require active intervention of the user - he can manage both the work of the system in order to achieve results, and modify virtual objects. According to this feature all systems are divided into:

1. Autonomous. They do not require user interaction. The objective of such systems is limited to providing information about the objects. For example, such systems can analyse objects in the field of human and give background information about them. Such systems, for example are used in medicine. Gait Aid system [9] for people with disorders of the musculoskeletal system. It is the use of virtual objects providing additional information about the brain that helps coordinate movement.
2. Interactive. Such support interaction with the user. On the different actions the user gets a different response. In such systems, the user must give input to the device. The touch-screen mobile phone, a tablet or a special manipulator can be used as such a device. Choice of input devices is dependent on the specifics of the system. In the case of simple actions with a virtual object, a simple pointing device should be enough. If, however, the aim is to simulate real processes and perform complex manipulations

with objects using special manipulators having a different number of degrees of freedom is required. An example is the PHANTOM device [10].

Interactivity is expressed to different degrees. There are systems that allow the user to actively change the virtual environment [10]. This is usually simulation system of any real action. They are used when it is impossible to use real objects, such as specialized medical simulators that allow novice physicians to practice the necessary skills [11].

There are other systems where the user does not need to change the virtual environment. Instead, the user selects which virtual objects he wants to see. The user also has an ability to manipulate virtual objects, not at the level of structure, but at the display level, i.e. used, for example, basic functions like transformation, rotation and displacement, etc. This group may include various architectural systems [12], that allow one to see how really a new facility or part thereof will fit into the existing environment, as well as navigation and geographic information systems. Such systems can display parts of objects of interest hidden by other buildings, additional information on the selected objects, etc.

In terms of augmented reality, systems of mobility can be classified as:

1. Stationary. Systems of this type are designed to operate at a fixed location; movement of such systems may result in partial or full suspension of their working capacity.
2. Mobile. Systems of this type can easily be moved; often such movement is the basis of the functions which they perform.

Affiliation to one or another type is determined by the functions of the system. For instance, surgical table simulator should not be mobile, because of its task - to recreate the human special conditions as close to real. At the same time, a navigation system should be as mobile as possible so that it can move together with a vehicle or a person, without creating additional costs for its movement.

3.3 Augmented reality hardware

Main devices used in augmented reality systems are displays, input devices, tracking devices and computers. In this chapter we review the most important device in our system – display.

3.3.1 Displays

There are three main types of displays used in augmented reality: head mounted displays (HMD), hand-held displays and spatial displays (projectors).

HMD is a device that is attached to the user's head, or to a special helmet, which puts the real image and virtual environment before the user's eyes. Video-transparent systems are more demanding than optical-transparent ones since they require the user to wear two chambers on the head, and therefore data from both cameras to display a "real part" of augmented scenes, and virtual objects. At this time, optical-transparent systems use half the silver mirror technology that allows looking at the real physical world through a special "lens" that imposes additional graphical information. The scene, as well as the real world in such a system, is seen more naturally. On the other hand, in the video-transparent systems, augmented reality is initially aligned with the real image, which gives one more control over the outcome. Thus, it also gives control over the time in which the final stage can be achieved by synchronizing the virtual image to display it. In the embodiment, the optically transparent displaying of the real world may not be delayed, but this leads to a delay in the introduction, graphics and image processing.

Under the hand-held displays refers a small computing devices with a display, which the user can hold in her hands. They use video-transparent techniques to overlay graphics on the real environment, as well as use additional sensors, such as a compass, GPS and accelerometer. Marker detection systems are often used for the implementation of augmented reality systems such as ARToolKit, or methods of computer vision, such as SLAM.

Spatial Augmented Reality (SAR) uses video projectors and optical elements, holograms, and other technologies to provide graphic information directly to the physical objects, without requiring the user to wear or hold the display [13].

Table 3. 1 shows comparison of various types of displays. In the table we compared three types of displays: HMD, Handheld and Spatial. We reviewed what type of technology use each type of displays and main cons and pros.

Table 3. 1 Comparison of different types of displays

Type	**HMD**		**Handheld**	**Spatial**		
Technology	Video transparent	Optical transparent	Video transparent	Video transparent	Optical transparent	Direct supplement
Advantages	Visual control, synchronization of the virtual and real environments, waiting video processing from cameras	Half silver mirror technology, the natural perception of the real environment	Portable, powerful, combination of many sensors	Price, can be adapted to the standard equipment	The most natural perception of real objects	Display directly on the surface of a physical object
Disadvantages	It is necessary to fix the device on the head, unnatural perception of real world	The time delay, jitter of virtual elements	A small display, weight (for tablets)	It does not support mobile systems	It does not support mobile systems	Do not depend on the user (everyone can see the same thing)

3.3.2 Input devices

There are many types of input devices for AR systems. Some systems use gloves. Others use wireless bracelet. In the case of smart phones, the phone itself can be used as a pointing device, for example, in Google Sky Map app on your Android phone requires the user to point the camera in the direction of a star or planet, which he wants to learn. Selection of input devices is largely dependent on the type of system being developed and selected display type. If an application requires that the user's hands are free, the input device will be selected accordingly similarly, if the system uses a portable display, developers can use as an input device - touch screen.

3.3.3 Tracking devices

Tracking device is a digital camera or other optical sensor such as GPS, accelerometer, compass, wireless sensors, etc. Each of these technologies has varying degrees of accuracy, and largely depends on the type of system being developed.

3.3.4 Computers

Augmented reality system must have a powerful processor, sufficient RAM and video memory to process the images from the camera. With the development of technology and the emergence of new portable and powerful, at the same time, devices such as smartphones and tablets, the power shortage problem is relegated to the second plane.

3.4 Conclusion

In this section we reviewed concept of augmented reality application and required devices. This information helps to determine view of our application. Based on the theoretical information that we have reviewed, we defined the next main characteristic for application:

1. Visual type
2. Optical type
3. Interactive
4. Mobile type
5. Use HMD as display

Having these criteria for the application, we can begin to move forward to the practical part. In addition, these criteria will serve as guidance for us during writing this work.

4. COMPONENTS OF THE SYSTEM

In this section we consider all required components for our application. The main components are HMD, camera, motion recognition system and PC. In addition, we review alternatives for each component and find the optimal solutions.

4.1 HMD

HMD or head mounted display, is a display device, worn on the head or as part of a helmet, that has a small optic display in front of each eye; the picture on the screen always corresponds to the direction in which the person looks. All this makes the helmet suitable for games or immersive panoramic video – a movie that goes on around you and you can look around in life.

In our laboratory class there is only one available type of HMD at the moment – Oculus Rift DK2.

Oculus Rift is certainly the most famous virtual reality helmet that gave rise to the current boom of VR-technologies and HMD. Oculus Rift is connected to a computer via DVI or USB and also carries out tracking of the user's head [14].

This device is a very important part of the whole system and is a key element in the creation of augmented reality applications.

4.2 Camera

In any application of augmented reality there must be a camera to read images from the environment. To increase the effect of presence, it was decided to use a stereo camera, but it was also decided to consider a possibility of using monocular web camera. It is important to note that webcam is present in every house, which is not true about a stereo camera. Stereo cameras are a very expensive component, and are currently quite rare.

A stereo camera is a camera that has two lenses about the same distance apart as one’s eyes and it takes two pictures at the same time. This simulates the way we actually see and, therefore, it creates the 3D effect when viewed [15]. Also important to note, that the stereo

vision is a very complex process and to catch stereo image should be respected significant aspects like optical calibration, focus angle and etc.

We chose two popular stereo cameras on the market now:

1. Ovrvision Pro
2. Zed camera

Also for experiments we used a usual web camera represented by Logitech HD Webcam C615. This camera has a high HD resolution and is easy to integrate into any plugin or engine without special software.

The table below (see Table 4. 1) compares our cameras.

Table 4. 1 Camera comparison

Camera Type/ Description	**Ovrvision Pro**	**Zed Camera**	**Logitech HD Webcam C615**
Video (60 fps)	1280x800	1280×720	1920 x 1080
Field of view	Hor.ang.115° Ver.ang. 105°	110° max.	- -
Depth Range(m)	-	1-15 m	-
Size(mm)	100x75x45	175x30x33	-
Weight	65g	159 g	-
Compatible	Windows, Linux, OSX	Windows, Linux	Windows, OSX
Oculus rift mount	+	-	-
Price without shipping(euro)	370,50	400,89	90,00

For our purpose, we decided to use Ovrvision stereo camera, because this camera has a special mount for Oculus Rift and plugins for Unity and UE4 (chose of the latter two is motivated later).

4.3 Motion recognition system

In this project we would like to transmit body position information to application. Therefore, we have to choose a system for body motion recognition in order for the person to be able to transmit commands to the program. As a matter of fact, body motion recognition is a complex subject and is a matter of a separate project, this in this paper we will review the core features only.

In the following section, we review different motion recognition systems, their corresponding issues and challenges and, as a result, choose one system for our application; the process of integration of the chosen motion recognition system into the overall system is discussed in later sections.

4.3.1 Overview of notion recognition systems

Today, computer games are controlled by different controllers that are responsive to touch. These systems can be divided into two types:

1. Contact
2. Contactless

The contact motion recognition system is a system for body motion recognition, where sensors are fixed on the body as a suit.

The contactless motion recognition system is a system for body motion recognition, where sensors which are fixed away from the body. Good example of this type of system is Microsoft Kinect.

If we have a look at the market, we cannot find finished products of contact motion recognition systems like a suit. No doubt that there are some prototypes and controllers like PlayStation Move and Nintendo Wii. But we cannot buy and use the prototype like sensor based skeletal tracking system. Therefore, we do not review this type of system for our application and reclassify PlayStation Move and Wii to the contactless group. If we consider contactless sensor for motion recognition, we can find a lot of different finished product like Kinect or PlayStation Move.

4.3.2 Review of alternatives for motion recognition

As was declared earlier, we should transmit information about body position into the augmented reality application. For this task, we decide to use a contactless motion recognition system. Below we consider existing options of such systems. To narrow the sample down we have to declare criteria. First of all, this system should be available in Estonia; possibility should be connectable to PC and should have good accuracy of motion recognition. Also it is very important to consider price and availability of plugins for game engines.

Kinect

It is a peripheral device for Xbox 360, developed by Microsoft. Kinect allows the user to interact with the console without a game controller - through verbal commands and body posture. Software, using a 3D-camera, provides a complete three-dimensional detection of body movements, facial expressions and analyses voice of the player. Unique microphone allows Xbox 360 to perform sound source localization and noise suppression, making it possible to speak without headphones and microphone [16].

In Estonia, the controller is sold with new consoles, such as Xbox 360 Slim, but is also available separately at a price of around 26 EUR [17].

Kinect V2

Kinect 2.0 is an updated game controller for Xbox One from Microsoft, which came to replace the first generation of Kinect. The second version of Kinect is based on the same technology, but unlike the original version of the controller, Kinect for Xbox One console allows to use resources directly from the controller, such as voice recognition system, a microphone and a video camera that can be used for recording and streaming video. The new version of Kinect includes extension of the field of view, addition of high definition, ability of the camera to support simultaneously six players, and player tracking heartbeat [18].

Price in Estonia is 146 EUR [19].

PlayStation Move and Nintendo Wii

These types of controllers for motion recognition are very popular in nowadays. But we are looking for systems which do not restrict motion. In other words, if your hands hold a manipulator, it will interfere with the full immersion in the application. At the same time,

these types of controllers track only your hand and do not recognize the motion of another part of the body. Therefore was decided to abandon this type of motion recognition system.

4.3.3 Conclusion

After we reviewed possible options of recognition system we obtained only two types of systems. Actually, it is not very wise to compare old and new versions of Kinect, because the new version will *a priori* be better. At the same time, in our laboratory, we have only the first version of Kinect; therefore, we will use this type of motion recognition system. In addition, second version of Kinect is more suitable for our project, because Microsoft released special plugins for UE4 and Unity engines. It means that it would be very easy to integrate Kinect into our application without additional classes in the game engine.

4.4 Personal Computer

In this section we elaborate on the main hardware part of a PC for the augmented reality applications to work properly.

Important part of a PC for virtual reality and augmented reality application is the graphics card. To achieve smooth graphics on the PC, the card usually needs to support the game at 60 frames per second. This means that the computer needs to redraw 60 images per second. Reduce this setting, and one will see the image in all cells and brakes in visual effects. In virtual reality 90 frames per second is the optimal requirement. Since the picture is displayed in high resolution close to the eyes, any delays or ripples in the movement will be increased. And even worse, gameplay will have worse response time to the movements, which eventually can cause nausea. Minimum requirements for Oculus rift are NVIDIA GeForce GTX 970 or AMD Radeon R9 390.

Other components do not have such a strong influence and standard average quality components should serve the purpose well.

4.5 Conclusion

In this part we have considered the prerequisites for the application. Based on the review of alternatives, we chose the following components:

1. Camera – Ovrvision Pro
2. HDM – oculus Rift DK2
3. Motion recognition system – Kinect

Personal Computer – computer with graphics card NVIDIA GeForce GTX 970 or AMD Radeon R9 390 or better.

5. SELECTING FRAMEWORK FOR DEVELOPMENT OF THE APPLICATION

Before starting the development, is necessary to choose the framework for application (a set of libraries). In choosing the framework it is very important to consider required criteria for the application. Each application has own direction and purpose, therefore, it is very important to determine the main purpose before developing a version of the final product.

In our case, we have the following criteria:

1. Possibility to create augmented reality application
2. Possibility to integrate virtual objects
3. Possibility to integrate motion recognition system
4. Real time updating – interaction with system
5. Cross-platform
6. Free

The final product should have a game form – an augmented reality game.

Based on the above mentioned criteria and on the final goal, we proceed to review and analyse suitable frameworks. First of all, we have to review main game frameworks and their possibility of integration with other frameworks like OpenCV, Vuforia and ARToolKit.

5.1 Game Engines. Review of alternatives

At the moment, there are many game engines, from the GameMaker and Marmalade to Unity and Unreal Engine, and this is due to the fact that companies make it easier to write your own engine, which meets all requirements, rather than use an unfamiliar game engine, to study its interface, its programming language and etc. But now the game engines have evolved and many indie developers choose one or the other engine, in order to save time and money.

In this section we consider most popular game engines and choose the optimal solution for our case.

5.1.1 Unity 5

Unity is a cross-platform game engine developed by Unity Technologies and used to develop video games for PC, consoles, mobile devices and websites. Unity can be downloaded free of charge. If revenues are more than $ 100,000, one will need to buy the PRO version for $ 1,500 or $ 75 per /month. Currently, Unity has the largest community. Unity 5 uses C # and JavaScript. Unity has a powerful store - Unity Asset Store (just for this store alone, one can safely select Unity), which has absolutely everything from textures and 3D models to finished projects. None of the competitors can boast a range as wide as that. Asset Store brings Unity in leaders of market [20].

Advantages:

1. Powerful store Unity Asset Store
2. Largest community (lessons, helpdesk, forums)
3. Free start license
4. A large number of additional plugins

Disadvantages:

1. Focus on indie-games
2. Troubles with multiplayer mode
3. High requirements for computer
4. Free version does not allow to lay out the project for all platforms
5. Set of tools is limited in the free version
6. It is necessary to know programming languages

This engine has strong advantages for a beginning developer, but behind beautiful advertising stands a huge number of unresolved problems, such as the balance of detailing and performance. But based on the fact that it is necessary to develop applications that require integration of various additional plugins – Unity is a good engine option.

5.1.2 CryEngine 3

CryEngine 3 is a game engine developed by Crytek. The CryEngine, unlike other game engines, aims at creation of games for PC and consoles. Crytek made a very powerful engine, which delivers the best graphics of all the engines that are on our list. There is dynamics, and light, and shade, and misting in real time, and control of the level of detail of the landscape. If

you need an external component for your game, the CryEngine will make it better. Faces and characters obtained in CryEngine are truly spectacular.

CryEngine is free for non-commercial purposes and for learning. But now this engine will be distributed on a "pay as you wish" model, that is any developer can get this product in full, and not pay a cent of its developers, on the condition that he would be at odds with his conscience. Even if you make a commercial product and earn money on its sales, Crytek will not require any royalties from you. [21]

Advantages:

1. Absolutely free
2. Plugins for HTC Vive, Oculus Rift and PlayStation VR
3. Official lessons

Disadvantages:

1. Difficult for beginners
2. It is necessary to know programming languages

At the outset of this work, CryEngine was chargeable and therefore, this engine dropped out of the contest. But if we take into account that right now this engine is absolutely free - it becomes rather attractive for developers.

5.1.3 Unreal Engine 4

Unreal Engine 4 – game engine created by Epic Games. Unreal Engine 4 is the most popular game engine to create AAA-films (the highest level of films) and projects. This engine has high graphics capabilities. With Unreal Engine 4 one can develop games for the PC, Mac, and console, IOS, Android and Windows Phone. Unlike Unity, UE4 is a powerful tool for designing game levels directly into the engine; it has an incredibly comfortable Blueprint system (visual scripting), which is unique; it also has beautiful design of the engine and is intuitive to us. Of the entire game engine universe, Unreal Engine 4 is the most innovative. It combines high performance build, better graphics and ease of use. There is a very strong community which helps to solve all problems. Epic Games gives UE4 absolutely for free, the source code is open. All that is charges is 5% from the sale of a single copy. Epic team is working closely with the developers of VR technology, which means that now one can create games for PlayStation VR, Oculus, etc. [22]

Advantages:

1. Free
2. Blueprints
3. For all platforms
4. Easy to use
5. Additional plugins for augmented reality
6. Strong community

Disadvantages:

1. Not all additional plugins are available and working (concerning Unity)
2. Weak Asset store (concerning Unity)

5.1.4 Conclusion

In conclusion, it is tempting to say that Unreal Engine is a good free game engine, because of having the capability to use blueprints instead of coding, because it is a cross-platform engine and it is easy to use. But this engine has one big disadvantage compared to Unity - all new plugins come out first of all for Unity. As an example, we can review plugin for Ovrvision stereo camera and notice that firstly it was released as a plugin for Unity engine. A plugin for Unreal engine still does not work properly. In the Table 5. 1 are shown results of comparison.

Table 5. 1 Games engines, table of alternatives and criteria

Criteria/Engines	**Cross-platform**	**Availability of documentation**	**Programming languages**	**Additional services +AR and MRS (Score)**	**License type**
Unity 5	All platforms	Much	C#, JavaScript	Excellent	Free, Full version – 95$ per/m
CryEngine3	Windows PC, PlayStation 4, Oculus Rift, Xbox One, Linux PC	Few	C#/.net, C++	Bad	Free
Unreal Engine 4	Windows, OSX, Android, Linux, Oculus Rift, PlayStation, Steam, HTML5, Xbox.	Enough	C++, Blueprints	Good	Free, 5% from the sale of a single copy

5.2 Additional plugins for augmented reality application

Due to the fact that the game engine does not have built-in libraries to work with a video stream and create an augmented reality application, we have to find additional third-party plugins for that. In this section, we consider the following plugins to achieve this: OpenCV, Vuforia and ARToolKit.

5.2.1 OpenCV

OpenCV is released under a BSD (permissive free software licenses) license and hence it is free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for

computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Usage ranges from interactive art, to mine inspection, stitching maps on the web and advanced robotics [23].

This library is suitable for creation markerless and marker application.

5.2.2 Vuforia

Vuforia SDK - this is software for mobile devices, which allows one to create augmented reality applications. It uses computer vision technique to recognize and track simple flat images and 3D-objects in real time [24].

Vuforia SDK supports a variety of 2D and 3D target types, including markerless targets. Additional SDK features include localized occlusion detection using virtual buttons and images, performing target selection, and the ability to create and modify target set programmatically at run time [24].

Vuforia provides an API for C ++, Java, Objective-C, and .Net languages, also there is an extension for Unity game engine. Thus, the SDK supports both native for IOS and Android languages and at the same time allows the development of applications of augmented reality in the Unity, which are easily portable across both platforms [24].

5.2.3 ARToolKit

ARToolKit is a computer tracking library to create applications with augmented reality. For this it uses ability to monitor the video, i.e. the real state of the calculation and orientation of a camera relative to the physical square marker in real-time. When the actual position of the camera is known, virtual camera may be located at the same point and a 3D model is superimposed on a real marker. So ARToolKit solves two key problems in augmented reality: tracking the point of view and interaction with virtual objects.

This plugin is suitable to quickly create marker type augmented reality application [25].

5.2.4 Comparison of plugins

In this part of work, we reviewed additional plugins for game engines. All important information is summarized in the table below (see Table 5. 2). In the table are compared three plugins: OpenCV, Vuforia and ARToolKit. We determine the next criteria for comparison: cross-platform, availability of documentation, programming languages, additional services and type of license.

Table 5. 2 Additional plugins, table of alternatives and criteria

Criteria/ Alternatives	**OpenCV**	**Vuforia**	**ARToolKit**
Cross-platform	iOS, PC, Android, Linux, Unity, Unreal Engine	iOS, Android, Unity	iOS, Android, Unity,
Availability of documentation	Much	Few	Enough
Programming languages	C/C++, Python, Java, Ruby, MATLAB, Lua	C++,Java, Objective-C, .Net	Java, Objective-C
Additional services (Score)	Excellent	Good	Good
License type	Free	Free + Commercial SDK option	Free + Commercial SDK option

5.3 Conclusion

In this part of the work we have to choose appropriate frameworks for creating augmented reality applications. Clear objectives were set and with accordance with these criteria and the main goal, more suitable engines and plugins for our application were reviewed.

First of all, we have to choose the engine as base of application. In our opinion and based on the facts from research, we decide to use Unreal Engine 4 as a game engine, because this engine is the optimal solution in this case. This engine is free to use, cross-platform, suitable for working with virtual 3D objects and interacts well with extraneous plugins like OpenCV and ARToolKit. Also, friendly interface makes learning easier and quicker.

It is worth mentioning that Unreal Engine is inferior by many characteristics to Unity, but the inability to use Unity in commercial projects and pared down functionality have determined the choice of engine.

As additional plugins for engine we reviewed OpenCV, Vuforia and ARToolKit libraries. As the game engine have been chosen as UE4, then only two plugins are suitable in our case: OpenCV, ARToolKit. To make a comparison of both plugins it was decided to use both for application and find the most suitable for augmented reality empirically.

6. VIRTUAL OBJECT CREATION

Virtual reality objects are a necessary feature of any augmented reality activity, because the virtual objects essentially "hold together" reality and virtually in the correct way.

During integration of virtual objects into augmented reality it is very important to observe the following aspects:

1. Scale
2. Similarity of physical shape
3. Quality of texture

Adhering to these requirements will help to create a realistic virtual object. It is very important, because we would like to achieve the similarity of the virtual objects with their real counterparts. In order to get a feeling of presence and full involvement integrated objects should not be evident in the environmental. Therefore, should we should pay particular attention to creation of virtual objects.

To create virtual objects special programs for creating and editing three-dimensional graphics and animation are typically used. There exists a large number of this type of software, but we consider only high-end types. We identify the main programs as:

1. 3ds Max
2. Blender
3. Maya
4. To select the best program, we set a number of criteria: free
5. Easy to learn
6. Possibility of texturing
7. Possibility to export objects to UE4
8. Cross-platform

6.1 Software for virtual object creation

After we specified the criteria, each program can be considered in more detail and, after that, we should be able to find the optimal solution for our augmented reality application.

6.1.1 3ds MAX

3ds Max has a large number of tools required for modelling a wide variety of objects and projects. This program is very complex, but 3ds Max is easy to learn, and the lack of any specific tool is compensated by a large database of add-ons - plugins that greatly extend the standard features of the application. 3ds Max is very popular software for artist and 3D-designers, but at same time, it costs 1,548.80 EUR per year. For students there is a 3-year free license [26].

6.1.2 Maya

Among professional 3D-artists this package is used more often than others. This three-dimensional editor has been adopted by major studios such as Pixar, WaltDisney, Dreamworks and others. The program contains everything one needs to create three-dimensional graphics. Maya allows to go through all the steps of creating 3D - from modelling and animation to texturing, compositing and rendering layering. This three-dimensional editor can simulate the physics of hard and soft bodies, calculate the behaviour of tissues and fluids, emulate effects; it allows to customize even hair detail of characters, or to create dry and wet fur and to animate hair.

In addition, Maya is very easy to operate. Cost of this software is about 800 EUR per year or 246 EUR per month [27].

6.1.3 Blender

Once again, existence of this program proves that free does not mean bad. One of the main advantages of the program is cross-platform. Blender is equally good and stable on Linux and Windows. In addition, the program can work even on very weak PC configurations, up to netbooks.

The program includes a large arsenal of tools for creating three-dimensional graphics. For example, Blender can handle particle systems that control weight of individual particles in texturing it can apply a guide for animating and using external forces, such as wind.

In addition, the program has a fluid simulator that offers the user a huge opportunity to create fluid-body effects, such as smoke or liquid. In real time, the user can calculate physical tasks, such as to simulate behaviour of soft bodies. The program allows to edit the surface to use and customize the tooling metabolite characters.

This program is free of charge and this is a very sizable advantage, but interface is quite difficult and one needs to spend time to get used to it [28].

6.1.4 Results of comparison

In order to more clearly present the results of the review of programs for creating virtual objects, the results are summarized in the table below (see Table 6. 1):

Table 6. 1 Graphics software, alternatives and criteria

Criteria/ Alternatives	**3ds Max**	**Maya**	**Blender**
Cross-platform	Windows, Linux, OSX	Windows, Linux, OSX	Windows, Linux, OSX
Availability of documentation/easy to learn	Much	Much	Much
Possibility to export objects to UE4/Texturing	Yes/Yes	Yes/Yes	Yes/Yes
Additional services (Score)	Excellent	Excellent	Excellent
License type	Free for students/1.548.80 euro/year	Free for students/1.548.80 euro/year	Free for students/1.548.80 euro/year

After reviewing the options and their comparison, it was decided to use 3ds Max for virtual object creation. This choice was made based on the following factors:

1. Free for students
2. Cross-platform
3. A lot of lessons for beginners

4. There is a possibility to integrate object into UE4

In addition, we would like to note, that in case of a commercial project, the choice would have been made in favour of Blender, because this software is free.

6.2 Own object creation

To create own virtual object one has to know some rules – Golden rules for virtual object creation for applications and games. The following list describes the process of creating a virtual object as well as provides useful hints for increasing productivity:

1. Clear picture in the mind of the required object
2. Correct dimensions for the object
3. Each part with texture should have been drawn as separate part – otherwise there will be troubles on the texturing stage
4. Virtual object should be similar to the real object

In addition, to increase productivity it would be advisable to have a look at lessons for beginners.

For my application I chose the object which will be similar to a crossbar for running.

The picture below (see Figure 6. 1) shows the virtual object without texture. Also, one can observe that every item is drawn as separate part.

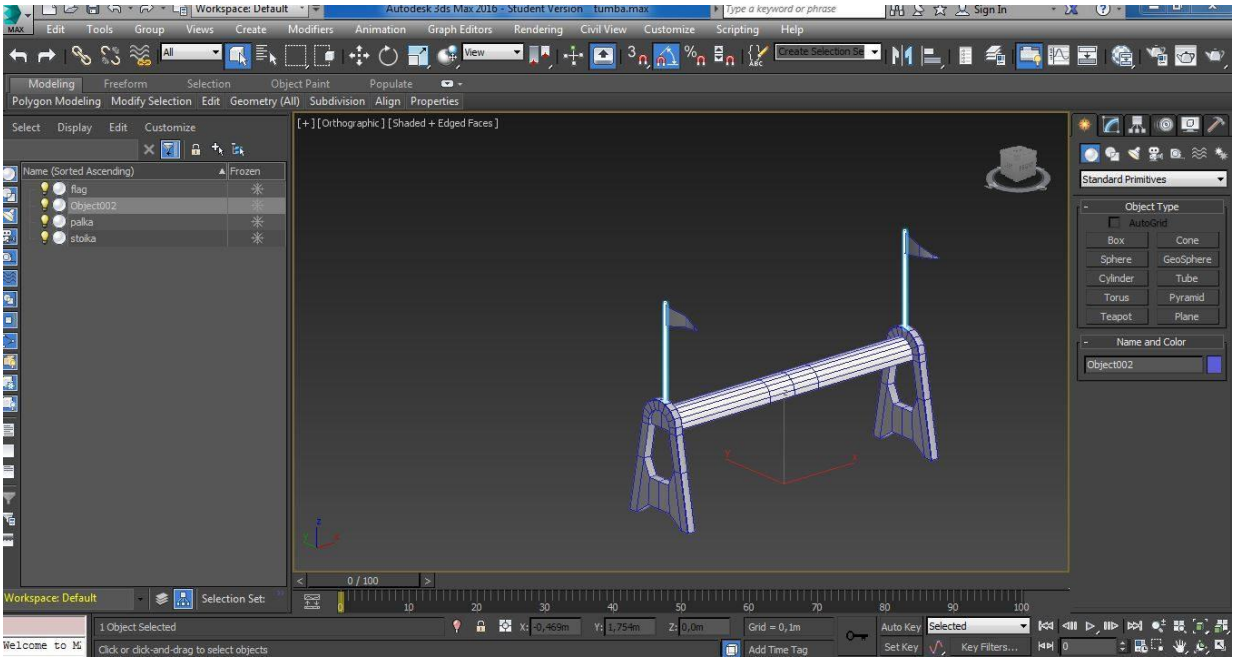

Figure 6. 1 3ds MAX, crossbar without texture

This figure (see Figure 6. 2) shows the object with realistic textures:

Figure 6. 2 Crossbar with realistic texture

Important part in virtual object creation is imposition of realistic textures on the object. Just compare the image with textures and without. As mentioned earlier, virtual object should be

as more realistic as possible for integration into augmented reality applications. This allows reaching the effect of immersion in the game.

6.3 Virtual object integration

The second step is related to object integration into the game engine. In our case we use UE4 engine and more suitable format for this engine is FBX. In UE4 we can use import function and, after that, find our object in the project folder. Also, you have to import all textures as well.

To link our object and texture, you have to use Blueprints. The next picture (see Figure 6. 3) shows this process:

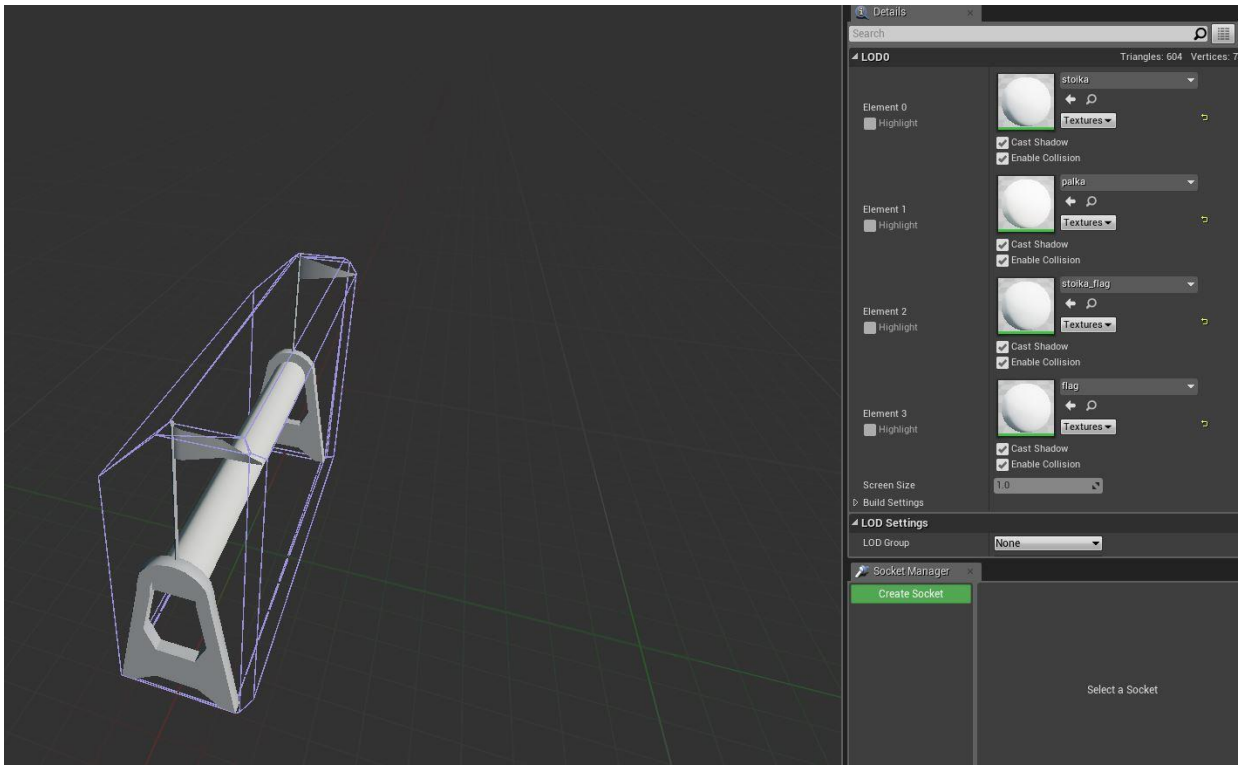

Figure 6. 3 Virtual object in UE4 without textures

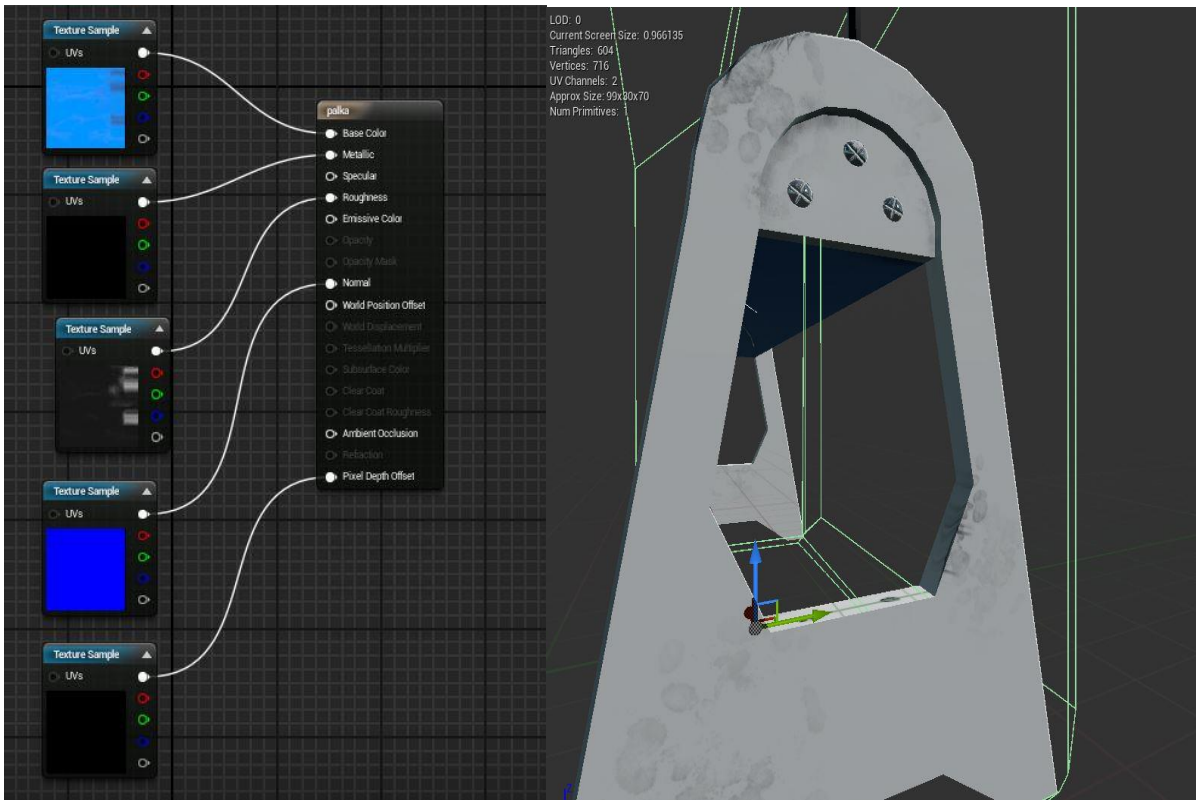

Figure 6. 4 Virtual object in UE4, Texturing and addtional components

At the beginning, our virtual object is grey coloured. For each part of the object we have to create a special texture blueprint and connect the textures. In Figure 6. 4 it can be observed that all exported textures like Base Colour, Roughness, Normal Colour, Metallic and Pixel Depth have been linked. To create the usual object, it is necessary to link only Base Colour, but in our case, when the goal is creating realistic objects, other textures re needed as well. These textures give our bar special properties like dirt and scratches or using different fasteners. All these details you can find the on the right hand side of Figure 6. 3.

In addition, it is very important to know that after importing the object will still not be the same as it was originally drawn. This problem arises from the fact that different programs read different textures in different ways. A good solution to this problem is to make textures inside the game engine. This process is not so convenient, but one can be sure that the textures look exactly as it was intended.

6.4 Conclusion

Virtual object creation is a very important stage in the process of augmented reality application creation, because the main goal of augmented reality application is a virtual object integrated into our environment. The object should have realistic textures and physical shape, in simple terms, it should be beautiful. Also, for virtual reality we can disregard some kind of unrealistic or absurd things, but for augmented reality requirements are more stringent, because we will compare the properties of the virtual object with the beauty of real objects.

In this section, we compared different software systems for creating and editing three-dimensional graphics and animation, chose a more suitable program for this project and reviewed virtual object creation and integration stages. In addition, we have identified some problematic nodes and found ways to solve the associated problems.

We also draw our object – the crossbar for running in the 3ds MAX software and imported it to UE4. We linked all previously drawn textures to the object inside the game engine. As the finished product we got a properly done virtual object for augmented reality applications.

7. APPLICATION DEVELOPMENT

In this section we will go through all steps of augmented reality application development.

We will use UE4 as game engine and different additional plugins which help to catch video streams from the camera and detect markers. The main goals of this section are to create the marker and markerless augmented reality applications, and compare them.

7.1 UE4 and Oculus Rift

For our application, we decided to use HMD - Oculus rift. The first step is to integrate Oculus into UE4. Unreal Engine 4 supports Oculus Rift through the use of a plugin. This plugin is included as part of the initial UE4 download. Unreal Engine 4 will automatically use Oculus Rift if it is plugged in and the plugin is enabled.

It means, that we can use official plugin and Blueprint block for own project. We found sample of OR First person level in the official forum [29].

In this template we have the following features of OR:

1. In game changing of important settings to the rift, screen percentage and distortion scale
2. Head Rotation Blend Spaces controlled by the HMD rotation
3. Semi-modular character so that once cannot see the inside of the head in game
4. Rift optimised global post processing
5. Character rotation based from head look
6. A context menu for changing locomotion, HMD settings and scalability settings
7. DK2 camera volume tracking and view

This template is free to use and we can take main features for OR in our application or create own level based on the template.

7.2 Marker method. ARToolKit plugin integration and use

ARToolKit is an augmented reality plugin for game engines like Unity and UE4. Augmented reality plugin for UE4 allows developers to use Unreal Engine 4’s Blueprint visual scripting

system to create multi-platform augmented reality experiences as seamlessly and as easily as possible [30].

One of the most difficult tasks of developing AR applications is to accurately calculate the coordinates of the point of view of the user in real time, so that the virtual images will precisely interact with objects in the real world. ARToolKit uses computer vision techniques to calculate the actual position and orientation of the camera relative to the marks on the map, which allows the programmer to overlay virtual objects on the map. Fast, accurate tracking ARToolKit aimed at rapid development of new interesting applications of AR.

After adding this plugin into UE4 plugin folder, we can use Blueprint nodes in our application. In our case, we would like to get the following features:

1. Transmit video stream to UE4
2. Detect marker using camera
3. Reproduce our virtual object on the marker

When we know required criteria for our application we can chose needed nodes from the plugin.

Required nodes:

1. ARToolKit framework nodes
    a. init
    b. cleanup
2. Markers node
    a. load markers
    b. get markers
    c. get relative transformation
3. AR camera
    a. get camera position
    b. get camera rotation
4. Web camera
    a. get camera frame
    b. get camera resolution
    c. create dynamic material instance
    d. set texture parameter value

For this type of application we have to use augmented reality markers. We can use standard markers, but is more interesting to create own marker and integrate it into the application.

7.2.1 Markers for ARToolKit

We can distinguish two main principles of construction of augmented reality:

1. On the basis of the marker;

2. Based on the user's location coordinates or other coordinate systems.

By a marker we understand an object located in the environment which is analysed by special software for subsequent rendering of virtual objects. On the basis of information about position of the marker in the space, the program can accurately project the virtual object onto it, so that the effect of its physical presence in the surrounding area can be achieved. Using additional graphics filters and high-end models, the virtual object can be almost real and difficult to distinguish from other elements of interior or exterior.

Often, a marker is a sheet of paper with a certain special image. Image type may vary but in any case it depends on the image recognition algorithms. Generally speaking, plurality of markers is wide: they can be geometric shapes or simple shapes (circle, square), or the objects in the form of a cuboid, or even eyes, hands and faces of people [31] (see Figure 7. 1).

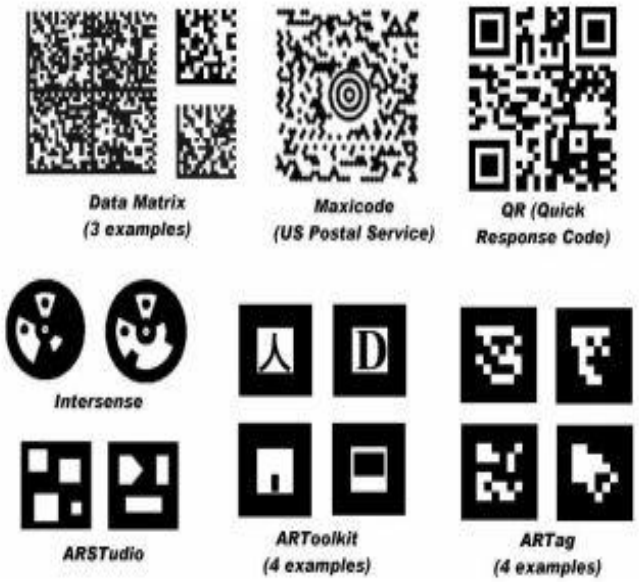

Figure 7. 1 Different types of markers [30]

Markers detecting

The important part of recognition of markers is computer vision. Computer vision is fundamental for development of augmented reality technology, and especially for the use of markers. The main direction of the discipline is analysis and processing of images (including video stream). Computer vision algorithms allow to highlight key features of the image (the corners of the boundary) to search for shapes and objects in real time and to perform the 3D reconstruction using several images and much more [31].

In the field of augmented reality computer vision algorithms are used to find special markers in the video stream. Depending on the task specifically formed images and faces can act as markers. After finding the marker in the video stream and computing its location, it is possible to construct a matrix for projecting and positioning virtual models. By means of the markers the virtual object can be superimposed on the video stream so that the effect of presence can be achieved. The main difficulty is to locate the marker to determine location in the block and to project a virtual model onto in [31].

The main methods of markers detection are: method of contour analysis, template matching, feature detection and genetic algorithms. From the viewpoint of constructing augmented reality often the last two approaches are used. We provide a brief explanation of each of them below.

Genetic algorithms

Genetic algorithms are the heuristic search algorithms used to solve problems of optimization and simulation by random selection, combination and variation of desired parameters using mechanisms reminiscent of biological evolution [32].

In computer vision, genetic algorithms are used to locate the object of a specified class on a static image or video stream. First, you need to conduct training of the algorithm using two different sets of images:

1. "Good" - contain the desired object.
2. “Bad"- a false image without the desired object.

To train the system a large number of images is needed and the more of them there re – the better the algorithm will operate. For each picture a selection of various key features is made:

border lines, central elements. During the training stage, a statistical model is built, which is then used to find the object in the image.

An example of this approach is as an algorithm for face and eye detection on the video stream. Gradually training the algorithm can ensure better results of finding a given class of objects. However, the need for training just makes use of genetic algorithms quite problematic. For their good work requires a significant number of different images (like "good" and "bad"), and the construction of the classifier for each object may take a long time [33].

Feature detection

The concept of feature detection in computer vision refers to the methods that are aimed at calculating image abstraction and discharging it on the key features. These features may be in the form of isolated points and curves or related areas. There is no strict definition of what is a key feature of the image. Each algorithm realizes that under its (corners, edges, areas, etc.) [34].

Point features can be used to find a sparse set of corresponding locations in different images. The key point is a certain part of the picture, which is distinctive for a given image. What exactly is taken as a key point depends on the used algorithm.

Figure 7. 2 Feature detection [35]

To find point and following comparison use of three components:

1. Feature detector - searches for key points on image.
2. Feature descriptor - produces a description of the found points, evaluating their position through the description of the surrounding areas.

3. Feature matcher - builds correspondences between the two sets of points.

First, a detector searches for the key points (see Figure 7. 2) of the template (desired) image. These terms are described later by a descriptor. This information is stored in a single file (or database), so that not to perform the process again. In order to establish a correspondence between the key points and descriptors used matcher [34].

It is natural to assume that different algorithms have different speeds and efficiency. In terms of their applicability to building an augmented reality application, only those that show high rate of work at a sufficiently good quality of tracking positions of key points must be used. Otherwise, we can get a noticeable lag in the filmed video [34].

To increase the speed of the feature points detection algorithms apply different ways of filtering points, in order to minimize their number and weed out the bad combination. Thus, it is possible to achieve not only increase in the speed of the algorithm, but also in the quality of tracking markers.

7.2.2 Own marker creation and integration

Markers are optical inputs to ARToolKit. About the Traditional Template Square Marker is one of several types of markers that ARToolKit recognizes and tracks in a video stream. A marker is simply a graphic image [36].

Square markers have only a few constraints:

1. They must be square.
2. They must have a continuous border (generally either full black or pure white). And, with the marker in the foreground, the background must be of a contrasting colour (generally, a dark versus a light colour or shade). By default, border thickness is 25% of the length of an edge of the marker [36].
    a. The final constraint is that the area inside the border, which we refer to as the pattern, must be rotationally asymmetric. The area inside the border can be black and white or coloured (and ARToolKit provides means to track with greater accuracy when the marker pattern is coloured).
3. ARToolKit supports recognition of a marker type referred to as a matrix marker which is made up of a 2D-Barcode Markers that is a form of two-dimensional barcode.

Matrix markers can speed up tracking when many markers are required in a scene, and when used with error correction and detection (EDC) offer increased resistance to one marker being misrecognized as a different marker [36].

We did our own marker (see Figure 7. 3), which satisfies the above requirements:

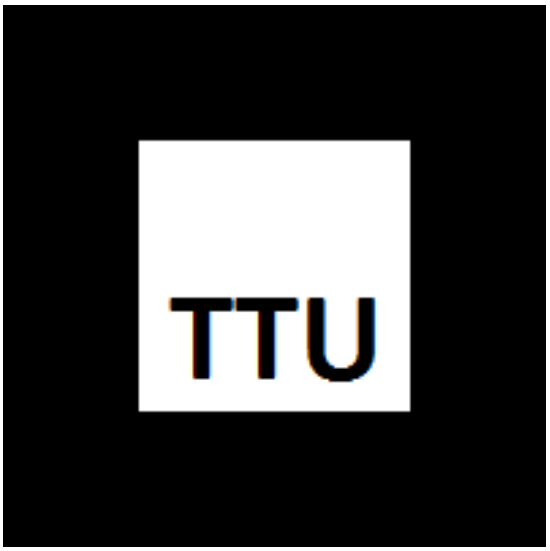

Figure 7. 3 TTU Marker

ARToolKit must be trained to recognize it. The output of the training process is a pattern recognition data file referred to as the marker's "pattern file." Pattern files enable ARToolKit to detect, recognize, identify and track new markers in a captured video stream [36].

In this section we reviewed step by step how to create own marker based augmented reality application using ARToolKit, created own marker and integrated own virtual object. In addition, we received empirical data using our application and will compare its performance with the markerless application.

In the figure below (see Figure 7. 4) you can find our object on the marker.

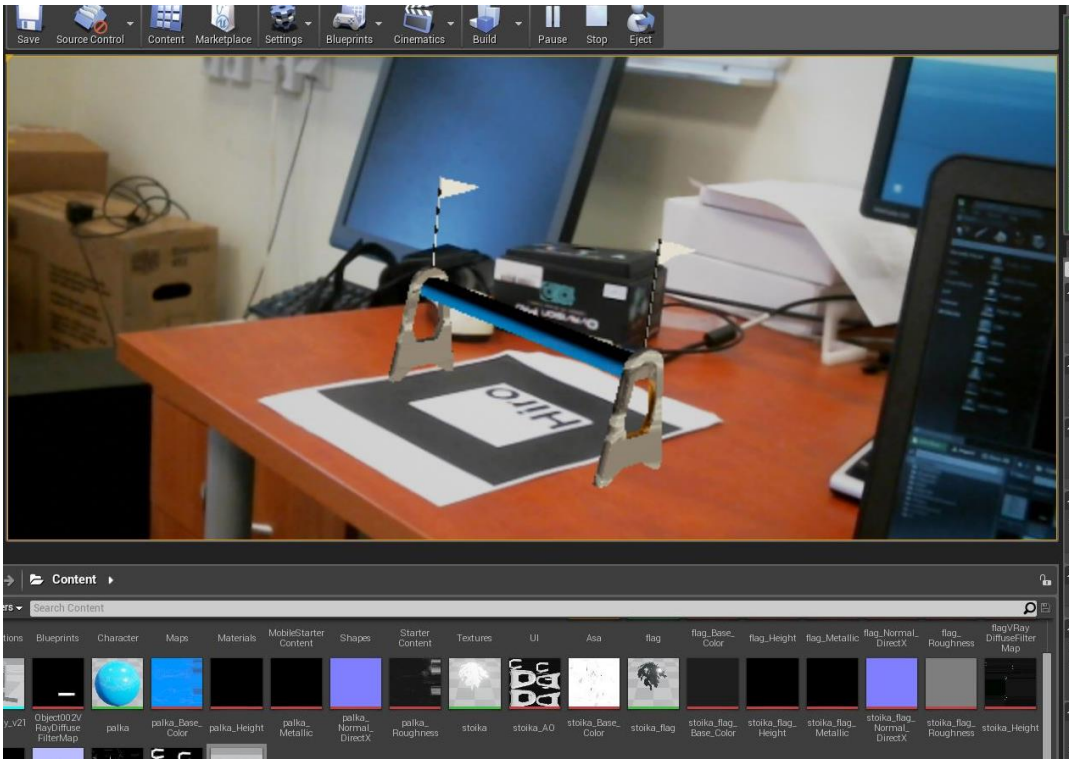

Figure 7. 4 Object on the marker

As was discussed before, one possible way to use this method is creation of AR application for laboratory works. We used as a sample MLS virtual model to integrate instead our crossbar. This equipment cost is 3.500 EUR. If we develop the application with a model of the system, we need only HMD and camera. The price is lower than to buy the system on the each class tool. The figure below (see Figure 7. 5) shows MLS object without texturing.

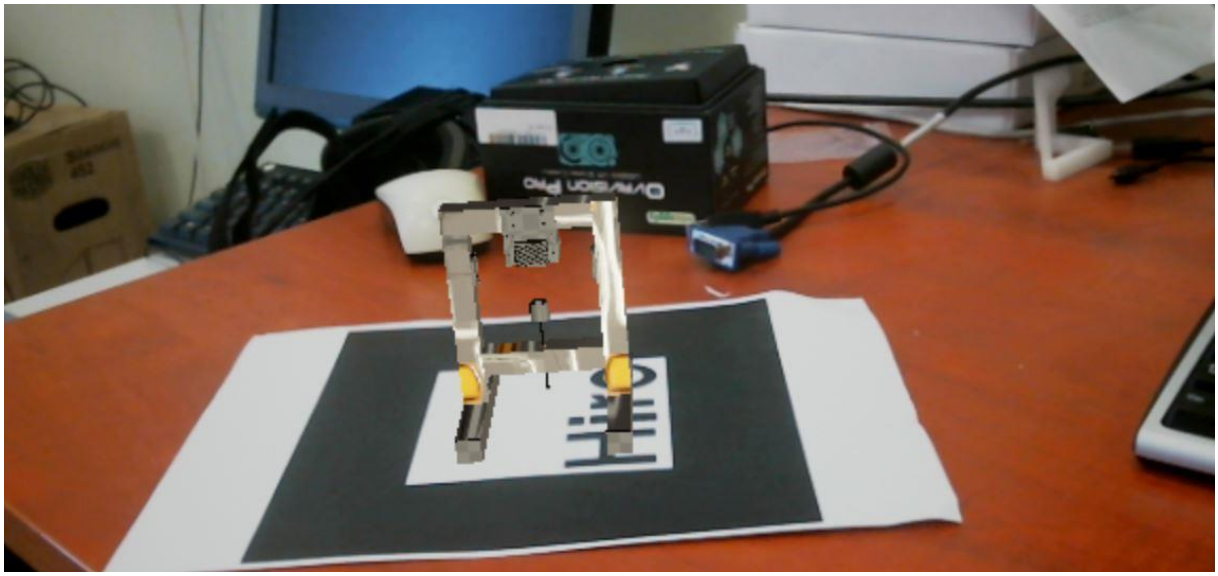

Figure 7. 5 MLS

During experiments with the application it was noticed that there are the following problematic points:

1. Scale of the object
2. Stability of object's position
3. Maximum distance between camera and marker
4. Dependency on illumination

The first problem we faced was related to the virtual object's scale. As was described above, virtual object is displayed on the marker which is read by camera. The marker has dimensions of an ordinary A4 sheet, and virtual object is attached to this sheet. It means that virtual object cannot be bigger than the marker sheet.

The second point is related to position of the object. Every frame our program reads information about the marker and analyses where the object should be placed. If we quickly change position of the camera our program does not have time to process information with the same speed and it moves the object abruptly. When camera position is fixed, then we do not have this kind of a problem, but for the game, where the character is constantly moving, it is not appropriate.

The third problematic point is maximum distance between the camera and the marker. This length is limited. Table 7. 1 shows results of our experiments with Logitech web camera and implications from changing the distance between the camera and the marker

Table 7. 1 Experiment results

Camera/Length	**0,5 m**	**1 m**	**1,5 m**	**2 m**	**2,5 m**	**3 m**	**3,5 m**	**4 m**
Logitech HD 615	Good	Good	Good	Good	Good	Bad	Very Bad	No

As you can see in the table (see Table 7. 1), maximum distance, at which the program recognizes the object, is 3.5 meters. Also, it is worth to notice that if illumination in the room was different the detection distance might have been better.

It means that we have to keep in mind this aspect when we create our own game, because maximum distance for marker detection is limited.

The last disadvantage of this method is dependence on illumination - the tracking result is affected by lighting conditions at room. Overhead lights may create reflections and glare spots

on a paper marker and make it more difficult to find the marker square. Shadows can be cast across the paper, breaking up white areas in the camera image.

7.3 Markerless method. OpenCV plugin

This section is devoted to creation of own augmented reality application based on the markerless method. For this purpose we use UE4 game engine and OpenCV library.

As was analysed before, OpenCV is an acceptable library for UE4 to catch video stream and transmit into the engine. In the previous section we created augmented reality application based on the marker method. In this section we will use OpenCV library as a plugin for UE4 to create an application based on the markerless method. Important to note, that OpenCV libraries can be used for marker detecting as well. This library is free to use.

7.4 Plugin integration into UE4

To use OpenCV libraries inside the game engine, first of all, we have to integrate the plugin. Unfortunately, no official OpenCV plugin exists for UE4 and unofficial plugins do not work correctly. In our case, it means, that we have to integrate required modules from libraries.

If we use cv::Mat frame variable in our UE4 project code, editor will underline these lines and show an error: *Mat: identifier not found*. This means that definition and function of these structures are not found. To solve this error it is required to connect a correct header file (.h and .hpp), where both are described. In our case:

- opencv2/core/core.hpp
- opencv2/highgui/highgui.hpp

Since we can have more then only two methods of OpenCV libraries, it is convenient to specify the base location where to look for these files, using #include.

- #include "opencv2/core/core.hpp"
- #include "opencv2/highgui/highgui.hpp"

The path to OpenCV will be determined in the project configuration, where the paths are specified.

In addition, there are two more places where it is necessary to specify the path. The first - Properties in the project, there is a section VC ++ Directories option - Include Directories. The second - a project build configuration file that has a name in this format:

- Source/%project_name%/% project_name %.build.cs

Settings from Properties use Editor to help with syntax errors and do not underlined in red the path to the included files. Settings from build.cs file are used when project assembling is being performed. So, you need to add a line to the constructor build.cs:

- Public Include Paths.Add ("D: / ... opencv ... / build / include");

Now we can try to build the project (Build). Using this method we can integrate needed parts for project for OpenCV library.

7.5 Markerless application concept and level creation in UE4

To create the augmented reality application based on the markerless method we have to come up with a concept of application. As the first step, we have to specify criteria:

1. Virtual object displaying without marker
2. Video stream from camera to game engine
3. Possibility to add new objects and interact with them during the game
4. Oculus Rift functionality

To achieve the criteria we have to divide our task into parts. The parts are listed below:

1. Concept
2. White blank level creation and integration of Oculus Rift functionality
3. Use OpenCV library to display video stream
4. Own virtual object integration and positioning

7.5.1 Concept

The concept is a very significant part in application development. Because ideas, which are incorporated into the concept, will determine the direction the development process with take.

Our concept is shown on the picture below (see Figure 7. 6).

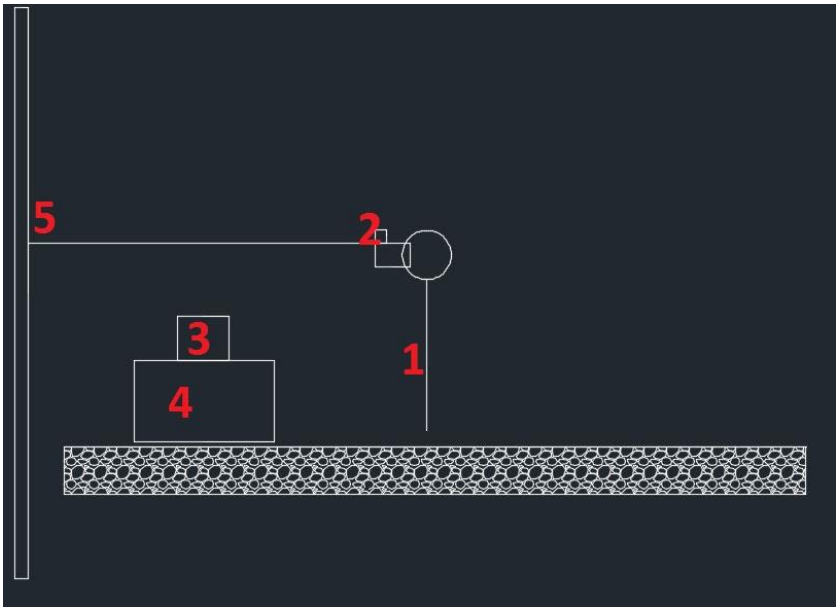

Figure 7. 6 Concept

1. Player And Character In The Game
2. Oculus Rift And Camera
3. Virtual Object
4. Real Object
5. Plane With Video Stream

Main ideas of this concept are:

1. To stream video onto the plane, which is in front of the player
2. To give position for the virtual object, which is connected with the real object or surrounding environment
3. Distance between the video plane and the player should be greater than the distance between the player and the virtual object

This concept allows us to see the virtual object and surrounding environment. Virtual object will be imposed on the video. In addition, virtual object will be fixed in the game, which in

turn will enable us to avoid same problems as in the marker method - object jitter, dependence on the movement of the camera and illumination.

7.5.2 Level creation and Oculus Rift functionality integration

For this task it will be easy to use Oculus Rift first person template and modify it for our needs. In this template we have blueprint classes, which spell out Oculus Rift functionality like head rotation, turns and tilts. After that, our character should have a solid footing, but the footing should be invisible, so that it does not interfere with the video plane. In addition, we add illumination.

7.5.3 Using OpenCV library to display video stream

In the plugin integration into UE4 section it was described how to integrate required OpenCV libraries into the game engine. But for video streaming on the plane, we have to create a special blueprint class. Now we will review blueprint class creation and will describe each block.

At the beginning we have to create the plane for video streaming and C++ class to communicate the blueprint to OpenCV. After that we can edit blueprint blocks.

Full code for C++ class is presented in the Appendix 1 page 80. This code is based on the example form UE4 wiki forum [37]. We modified this code for our purposes.

This class is used as a wrapper for the future UE4 blueprint class. It allows to specify device ID, target resolution and framerate of the camera, as well as to provide a dynamic texture and FColor array of the current frame's pixels and a blueprint native event that is called whenever the next webcam frame is available [38].

In the figure below (see Figure 7. 7) is shown all blueprint nodes. It will be easier if we divide this blueprint class into 3 blocks:

1. Event Tick
2. Event Begin Play
3. Event On Next Video Frame

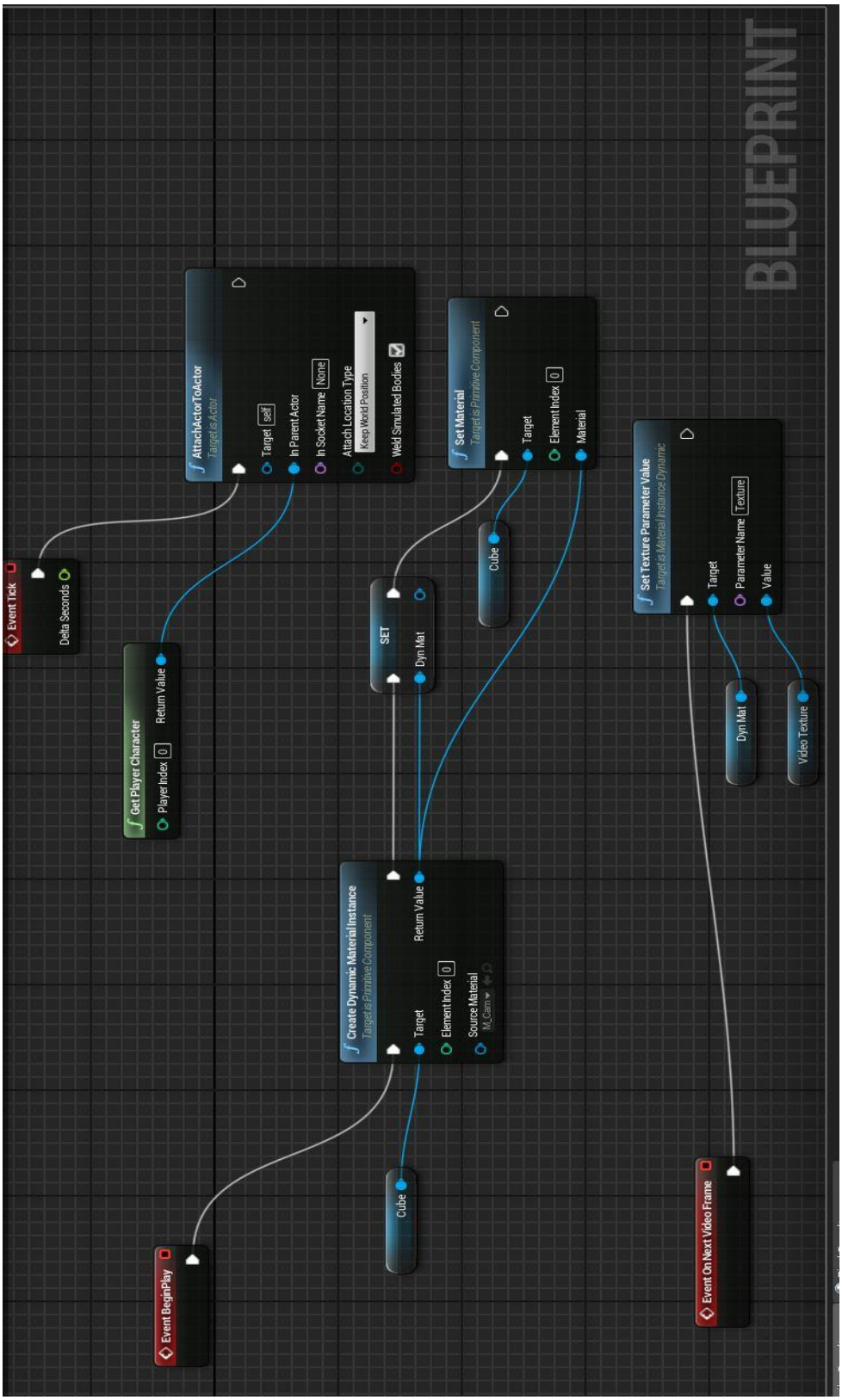

Figure 7. 7 Blueprint nodes for video stream capture

Event Tick block is a simple event that is called on every frame of gameplay. We use this block to give impulse every frame to the AttachActorToActor node' in simple terms, it means that AttachActorToActor receives a command to update itself every game frame. AttachActorToActor node is used for actors attaching between themselves. In our case this is necessary because we would like our plane to be always in front of the camera. As input for this block Get Player Character node, which determines parent actor (to which plane should be attached) is used.

Event BeginPlay block is executed when the Actor ceases to be in the virtual world. This node is connected to Create Dynamic Material Instance node. Dynamic Material Instance is an instanced Material that can be calculated during the gameplay (at runtime). This means that as you play, you can use the script (either compiled code or Blueprint visual script) to change parameters of your Material, thereby altering your Material throughout the game. Possible applications for this are endless, from showing different levels of damage to changing a paint job or to blending in different skin textures in response to facial expressions [39]. As input data we have our plane for video stream (we created this plane before). Create Dynamic Material Instance is connected with Set Material node through Set node. Set node is required to set a variable. The variable name is Dyn Mat. You can use random name for this variable, but type of the variable should be Material Instance Dynamic. After that we create dynamic material instance and set the variable we have to give the value to the plane. For this task we used Set Material node. Inputs are dynamic material as material, and our video plane as target.

Event On Next Video Frame is a custom event that we need to update texture parameter each time a new frame is received. The node is linked with Set Texture Parameter Value. Set Texture Parameter Value has two inputs:

1. Dyn Mat variable

2. Video Texture - needed to retrieve a reference to the webcam texture.

Set Texture Parameter Value is a standard node used to set material instance dynamic texture parameter value.

7.5.4 Our own virtual object integration and positioning

The last step to complete our application is to integrate our own virtual object. We will use the crossbar that we created before.

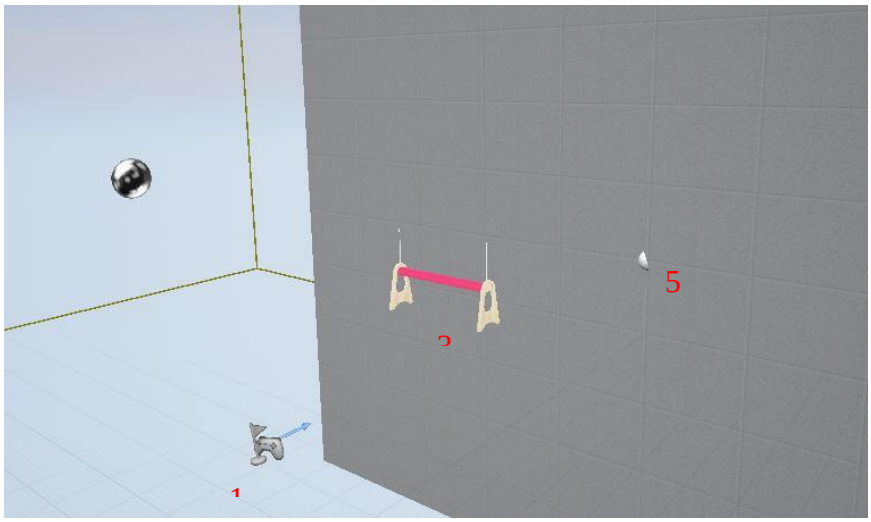

Figure 7. 8 Unreal Engine level

It is important to specify a coordinate system, which we will operate in for virtual object positioning. One of the possible options is to use position of the character as the origin point. In the figure above (see Figure 7. 8), that point is marked with a flag and a controller.

The figure below shows our object on the PC box (see Figure 7. 9).

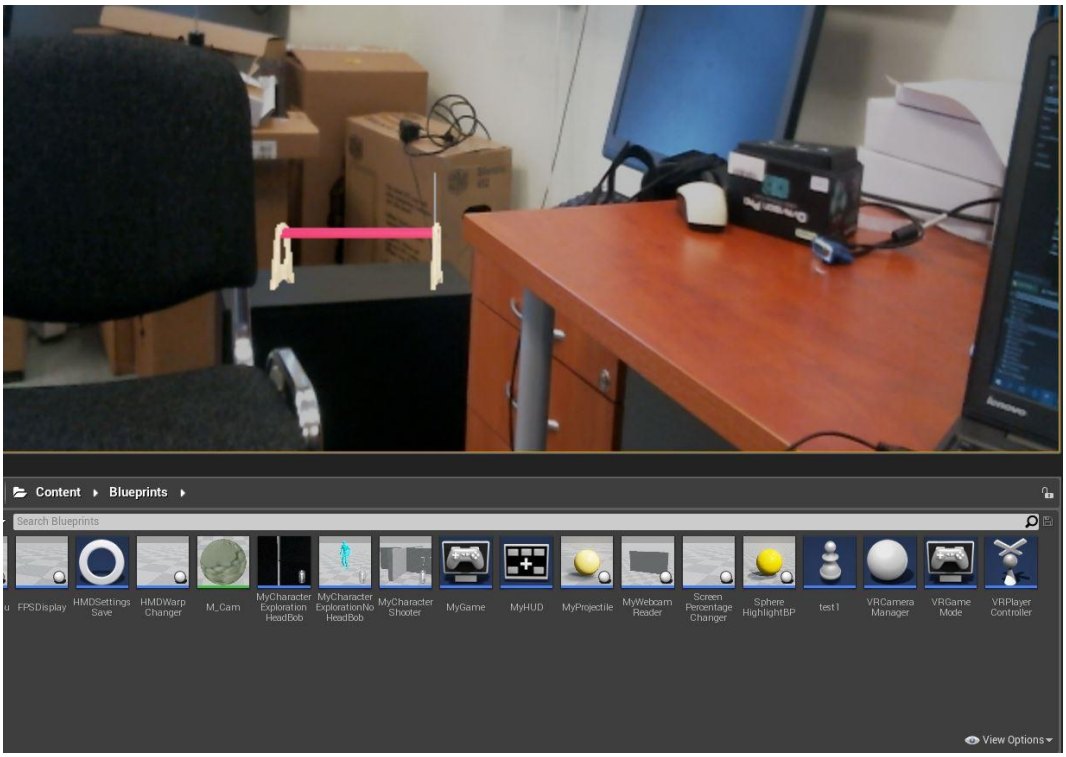

Figure 7. 9 Markerless method

7.5.5 Application testing and conclusion

Figure above (see Figure 7. 9) shows the image from the PC display. In this picture as background we transmit video stream from a web camera and in the foreground our virtual object is located. At first glance, one can detect that our object is not proportional to the environment. This effect occurs because of positioning of our plane with the video. Our plane is located at the distance of 3 meters from character's standpoint. The object is at distance of 1 meter. But, the main point is that our object is static and we do not notice any effects resulting from quick movement of the camera. Also, we avoid illumination influence on the marker detection, because we do not have any markers. One more meaningful advantage is the possibility to integrate many objects without creating markers for each one. Thus, markerless method allows to simplify the process of creating an application with a number of objects, because we do not need to create an own market for each object and to teach system to recognize all markers separately.

To solve the problem with object scale in real environment we have to think about the system of coordinates and about positioning of plane in the video stream. In addition, if the user looks at his hands the virtual object will overlap with them. This effect persists because we located

the plane with the video stream at a long distance from the character, longer then the virtual object. As a solution, we propose to create a special class to cut out our virtual object on the video plane. It means that the video plane will be directly in front of view (in front of character eyes) and we cut out the projection of the object on the video plane. It allows us to see the virtual object without overlapping with hands.

7.6 Augmented reality application for laboratory work

Based on the previous experience with augmented reality application (marker method and markerless method), we decided to review in more detail a possibility to use our application for laboratory work.

Laboratory equipment is very expensive and therefore many universities cannot afford buying interesting and useful equipment for laboratories. In addition, most often once piece of equipment is not enough for a group of students and as a result students can explore interesting equipment only through eyes. This sad state of affairs may be changed using augmented reality applications.

The idea is to display needed laboratory equipment on the table. To achieve this goal the following equipment is required: HMD device, web or stereo camera and PC with the application. As was shown in the previous chapter, we displayed MLS on the table using marker method augmented reality application. In addition, very significant part of the application, in this case, is a realistic drawn virtual object, which has the same shape as the real object. How to create realistic object we reviewed in the virtual object creation section.

To confirm the feasibility of using this type of application we decided to compare prices. First of all, we calculated price for whole setup that is required for an AR application. The total price is 690 EUR (OR DK2 –about 300 euro (sold out) and the latest version of OR is 600 euro, web camera – 90 euro, PCs are available in every computer class). To consider equipment for laboratory works we decided to take into account Inteco products like MLS and Tower Crane System. MLS price is 3,350 EUR and Tower Crane System price is 7,800 EUR. All prices were requested from official dealers.

Based on this comparison we can find that prices are very different. But, we should not forget, that virtual equipment can be compared with the real, because nothing will replace the real sense of a part and where we find virtual models of this equipment.

As a good solution for this problem, we offer to use AR applications with one piece of real components. Having one sample of real equipment, we can create own virtual model based on the real object. It will simplify the process of virtual object creation, because we can measure all parts. Also, having a real object will allow to provide student with real experience with equipment. At same time, we can use application to give student overview about object and to do basic experiments. Using this method we can reduce depreciation of equipment and provide more time to students to practice. This type of equipment is connected to MATLAB software and through this connection we can control devises using blocks and functions in MATLAB Simulink.

All in all, we can summarize that using AR applications for laboratory works is justified. Using this application can provide more practical experience to students and save money to universities. To achieve real results in this area, we should consider more closely a connection between AR application and MATLAB and try to implement it in the usual university life to collect feedback from students.

7.7 Conclusion

In this section we reviewed two ways of creating an augmented reality application. The first way is based on the marker method. To apply this method we have to use a special marker onto which the object will be projected. The method has the following advantages:

1. Plugins for game engines
2. A lot of information for learning

Disadvantages:

1. Virtual object scale, which is related to the marker dimensions
2. Processing time; dependence on the speed of camera displacement
3. Limited distance between the camera and the marker

4. Influence of illumination on the marker detection

The second way is use markerless method. We offered own concept and implemented it. In this concept we created a level with the virtual object and a plane with the video stream.

Advantages:

1. No markers
2. No influence of illumination
3. No restriction on the maximum distance between the camera and the object
4. No influence from the movement of the camera

Disadvantages:

1. Object scale selecting in accordance in environment
2. Object overlap the video plane

In this section we compared two methods and based on this comparison we can argue in which type of applications which method should be used. A more suitable method for laboratory work is the marker method. In this case, we do not have many objects, character activity is low, the distance between the camera and the marker is small, and illumination is uniform.

The second method of augmented reality application, the markerless method, is more suitable for games. Here very significant importance is attached to the speed of object processing and the number of virtual object inside the system. In addition, environment for games is not always well lit, which is another reason why we cannot to use the marker method.

In conclusion it is worth noting that the experience gained during exploring and implementing these methods allowed us to consider deeper principles of augmented reality. In addition, we gained real experience of AG reality application creation.

8. MOTION RECOGNITION SYSTEM

In this section we consider the process of motion recognition system integration into an augmented reality application. This system is required to interact with our application using our body. In our case, we would like to move virtual body in the game level, to change position of character in the game. It allows us to move closer to the virtual object.

In the motion recognition system overview we reviewed systems and chose the most suitable for us – Kinect. In addition, we will cover Kinect integration and Kinect V2 integration.

8.1 Kinect integration

First of all, it should be noted that Kinect is an obsolete version of Kinect V2. But if we have Windows 7 as operating system we cannot use the latest version of Kinect and we have to accept using the older version.

For this version of Kinect there are no plugins neither for Unreal Engine nor for other game engines, but we can use third-party programs to transmit the signal about our body positioning to the engine. One of these programs is FAAST or Flexible Action and Articulated Skeleton Toolkit.

FAAST is middleware to facilitate integration of full-body control with games and VR applications using either OpenNI or Microsoft Kinect for Windows skeleton tracking software. FAAST includes a custom VRPN server to stream up to four user skeletons over a network, allowing VR applications to read the skeletal joints as trackers using any VRPN client. Additionally, the toolkit can also emulate keyboard input triggered by body posture and specific gestures. This allows the user to add custom body-based control mechanisms to existing off-the-shelf games that do not provide official support for depth sensors. [40]

FAAST is free to use and is distributed for both commercial and non-commercial purposes. However, one must still abide by the licensing terms of any third party software for skeleton tracking (either OpenNI software or Microsoft Kinect for Windows). Please see the websites of these libraries for more information. [40]

This program will allow to connect our body to UE4. The principle is easy: we have to link body points from FAAST with a skeleton in UE4. The pictures below show skeleton from FAAST (see Figure 8. 2) and UE4 (see Figure 8. 1).

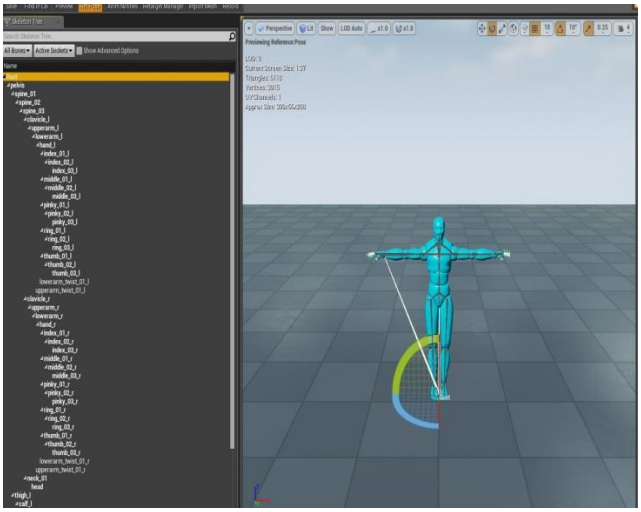

Figure 8. 1 Skeleton UE4

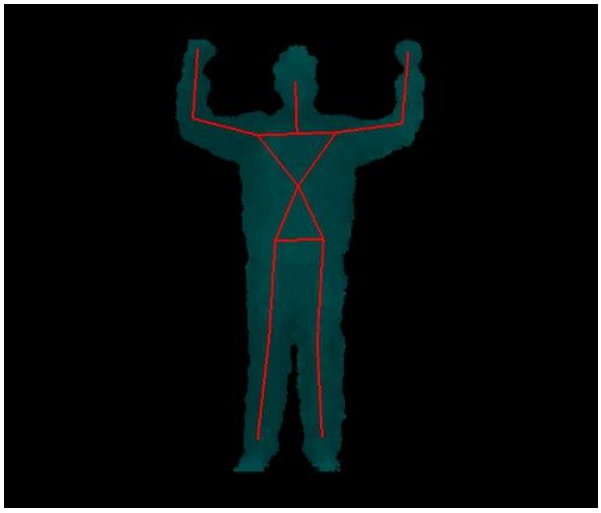

Figure 8. 2 Skeleton from FAAST [40]

Also we can declare special gestures, such as a keyboard button. For example, if the right hand is above the head – press W button.

This process takes a lot of time and effort. Also, after implementing it the application works with large delays. Skeleton points in the engine constantly jerk and change their positions due to the facts that:

1. FAAST has an error in reading the points from our body

2. Every small change and UE4 tries to reiterate

3. Movement is overdue due to having a delay in transmitting the signal.

In conclusion, we can truthfully say that motion recognition system based on FAAST application does not work properly for our purposes. Due to fact that we cannot control our virtual body in real time and have annoying spontaneous movement of limbs, we will not use this system for our application. Therefore, we have to find another solution for our application.

8.2 Kinect V2

In previous section we reviewed the process of integrating Kinect for motion recognition into UE4. This process took a lot of time and the result was unsatisfactory. Therefore we decide to have a look at Kinect V2. This system is newer and we there are ready to use plugins for UE4.

To integrate the plugin into the application one needs not more than 10 minutes. After downloading it one has only to copy-paste the required folder to the plugins folder. After this action, we can use Kinect V2 in our projects.

The Kinect V2 plugin provides the following possibilities [41]:

1. Joint location

2. Joint rotation

3. Joint delta position

4. Audio Volume/Beam angle

5. Camera frames

6. Avateering

In our case, we would like to retrieve information about body position and transmit it into the application. As we can see, this functionality is in the list above which means that there is no need to spend time connecting nodes manually; rather standard official plugin can be used rendering that unnecessary.

On the practical side, we have a block for each joint of the body. If we have this information we can refer to this block and get its coordinates. After that, we transmit these coordinates to our character in the game. But we do not need that, because we already made a proper special level as template in UE4.

It is worth noting that there is no Kinect V2 in our laboratory and a third party device was used for a restricted period of time to collect information. This device is not compatible with Windows 7 or older; the documentation states that it is compatible only with Windows 8 and 8.1. This important aspect should be taken into account at the beginning phase of the development of a system.

8.3 Conclusion

In this section we paid attention to motion recognition system and how to integrate them into our application. As was stated before, we looked at it rather briefly, without going into details. It was done due to fact, that motion recognition is a very interesting subject but is a matter of a separate investigation. Our goal, however, is to understand how to integrate a system for motion recognition into our application, but not to explore the system itself.

As result we have an overview of integration process of motion recognition system into UE4.

During our review, we came to the conclusion that Kinect V2 is more suitable for our project. The main advantage is that there is no need to use third-party application for integration, because an official UE4 plugin for Kinect V2 exists. Also, on the technical side, Kinect V2 has a newer sensor system and better algorithms for processing.

9. STEREO CAMERA INTEGRATION

Based on our research, we decided to use Ovrvision Pro stereo camera. This camera has the following advantages over competitors:

1. Weight is smaller
2. Smaller overall dimensions
3. Possibility to mount on Oculus Rift (special mount)

To use this camera, first of all, we have to calibrate it using instruction on the web page [42].

Camera Calibration is usually performed by multiple shootings of a calibration pattern; one can easily identify the key points which are known for their relative positions in space. Then the system of equations connecting the coordinates of the projection matrix camera and the position of the template points in space is compiled and solved. There are publicly available calibration algorithms, such as MATLAB Calibration toolbox. Also, OpenCV library includes camera calibration algorithms and calibration search pattern on the image. But in our case we have a standard application for calibration. Using this application we can calibrate our stereo camera in 5 minutes. The application is based on the chessboard calibration method. The proposed test object has the form of a chessboard (see Figure 9. 1) with a contrasting border, which makes it easier to separate the test subject from the background.

The developed software works according to the algorithm, comprising of 7 steps:

1. Registration of a set of images of the test object;
2. Image threshold filtering;
3. Recognition of the test object size and shape;
4. Remove background;
5. Determining position of four extreme points;
6. Calibration of the first and second cameras;
7. Calibration of the whole system.

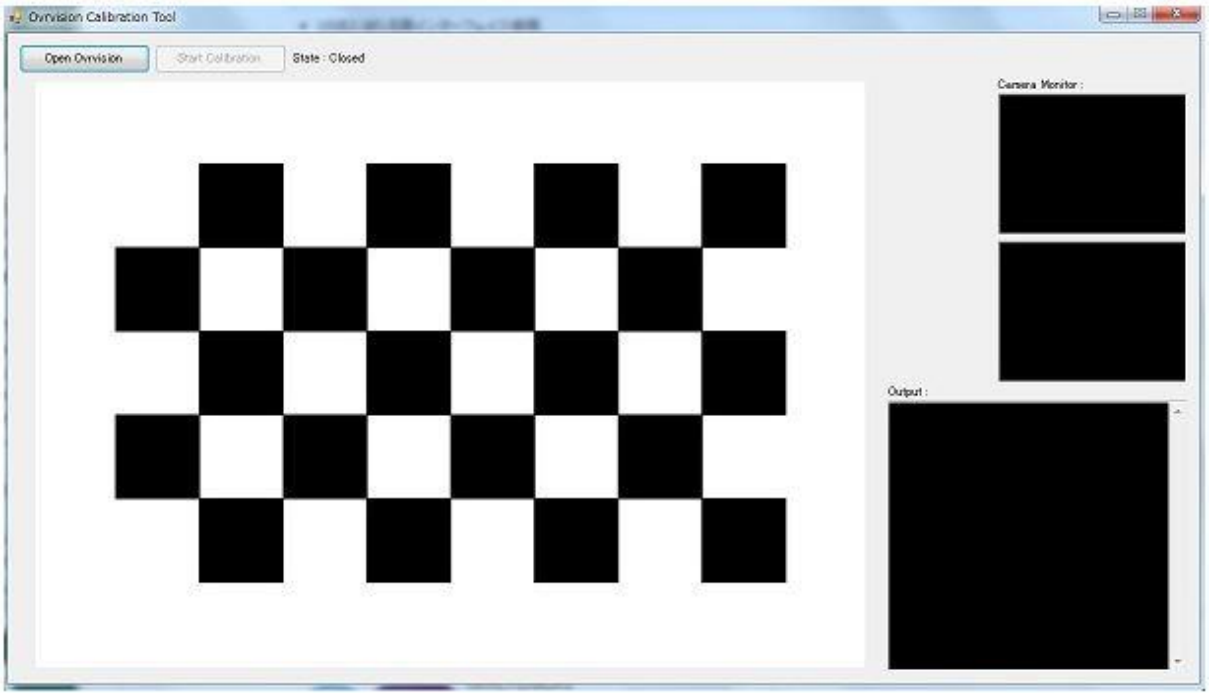

Figure 9. 1 Calibration

To obtain stable parameters for each camera and the stereoscopic systems 30 images of test object are required.

When we did an overview of the components, as major advantage to Ovrvision camera was the availability of a plugin for UE4. In reality, it turned out, that this plugin does not work and we cannot use for our purpose. This lamentable news directed us to explore possibilities of using OpenCV library for stereo camera integration into our project or to fix the plugin.

CONCLUSION

At the beginning of this thesis we have defined objectives. The main goal was an augmented reality application creation with motion recognition system and own virtual object. As secondary objectives, we identified the following:

1. Optimal choice of components and software
2. Creation of realistic virtual object
3. Integration of a virtual object into the application
4. Integration of a motion recognition system into the application
5. Find the most suitable method of augmented reality application
6. Unite all parts of the system

As result of our research and practical work we have achieved the following:

1. All components of the system have been reviewed and the optimal ones have been chosen
2. Virtual object with realistic textures has been created and integrated
3. Two different methods of augmented reality application have been explored and tested
4. Possibility of motion recognition system integration has been researched

During this work we went through all steps of augmented reality application creation and, as result, developed an augmented reality application. We reviewed two different methods for augmented reality application and compared them. After comparison we identified the pros and cons, designated areas in which they can be used and developed application based on the each method.

At the same time, we reviewed possibility to use augmented reality application for laboratory works. This is a very promising direction, because it can allow to reduce costs and increase availability of purchasing equipment for laboratory works.

In addition, we determined criteria for each component of the system, did comparison based on the criteria and chose optimal components for our project. This analysis will create a good basis for future works, because a comprehensive analysis of components for augmented reality applications was made.

We are faced with the following problems, which should be solved in the future:

1. Stereo camera integration
2. Object overlapping – markerless method
3. Object stability – marker method

Also, for the future, a plugin with the following functionality should be developed:

1. Web camera integration and operation
2. Possibility to use stereo camera, based on OpenCV
3. Possibility to use Kinect V2
4. Functionality of Oculus Rift

This plugin will help to avoid troubles with component integration and save time in future projects in the area.

KOKKUVÕTE

Käeoleva töö alguses said formuleeritud eesmärgid. Peamiseks eesmärgiks oli täiendreaalsuse (edaspidi TR) rakenduse loomine, mis toetaks liikumiste tuvastamist. Teised eesmärgid olid:

1. Optimaalsete komponentide ja tarkvara valik
2. Realistliku objekti loomine
3. Virtuaalse objekti integreerimine rakendusse
4. Liikumiste tuvastamise integreerimine rakendusse
5. Kõige sobivama objekti integreerimiseks meetodi leidmine TR rakenduse jaoks
6. Kõigi süsteemi osade ühendamine

Antud töö uurimuse ja praktilise töö käigus tahetakse jõuda järgmiste tulemusteni:

1. Kõik süsteemi komponendid said analüüsitud ja nendest valitud parimad
2. Realistlike tekstuuridega virtuaalne objekt sai loodud ja integreeritud rakendusse
3. Kaks erinevat täiendreaalsuse rakenduse meetodit said eksporditud ja testitud
4. Liikumiste tuvastamise süsteemi integreerimise võimalused said uuritud

Töö käigus said läbitud kõik TR rakenduse loomise etapid ja käesoleva töö väljundis on töötav TR rakendus. Töö käigus said vaadeldud ja võrreldud kaks erinevat TR meetodit (objektide integreerimiseks). Pärast võrdlemist tuvastati meetodite eeliseid ja puudusi, rakendamisvaldkondi, ja lõpuks sai loodud rakendus mis toetab mõlemad meetodid.

Samal ajal autor kirjeldas võimalusi TR rakenduste laboritöödes kasutamiseks. Autor jõudis järelduseni, et antud valdkond on väga perspektiivne, kuna võib vähendada vajalike seadmete soetamise kulusid ja samas suurendada selle kättesaadavuse (selleks, et seda katsetada ei pea seda soetama).

Samuti said valitud kriteeriumid süsteemi iga komponendi võrdlemise jaoks, võrreldagi ja lõpuks said valitud parimad komponendid käesolevas töös kirjeldatud süsteemi jaoks. Antud analüüs on hea alus teiste tööde jaoks, kuna selles oli tehtud väga ammendav komponentide analüüs.

Antud töös selgusid järgmised probleemid, mis peavad olema lahendatud tulevikus:

1. Stereo kaamera integreerimine
2. Objektide ülekattumine – markerita meetod

3. Objekti stabiilsus – markeriga meetod

Käesoleva töö autor hakkab edaspidi arendama antud tööd ja tahab saavutada järgmisi tulemusi plagina loomise vahendusel:

1. WEB-kaamera integreerimine ja rakendamine
2. Stereo kaamera kasutamise tagamine (OpenCV baasil)
3. Kinect V2 kasutamise võimaldamine
4. Oculus Rift'i funktsionaalsuse toetamine

Antud plagin aitab vältida probleeme objektide integreerimisel ja samas säästa aega sama teemaga seotud projektides.

REFERENCES

[1] D. S. Daniel Wagner, „ISWC '03 Proceedings of the 7th IEEE International Symposium on Wearable Computers,“ Washington DC, 2003.

[2] O. I. K. K. K. H. S. K. T. L. T. Heimo, „Augmented reality - Towards an Ethical Fantasy,“ *Ethccs in Science, Technology and Engineering*, p. 7, 2014.

[3] K. M. Y. I. G. K. Marios Bikos, „2.1 An Interactive Augmented Reality Chess Game using Bare-Hand Pinch Gestures,“ *2015 Internation Conference on Cyberworlds*, p. 4, 2015.

[4] „microsoft.com,“ Microsoft , [WWW].

Available: https://www.microsoft.com/microsoft-hololens/en-us. [04 30 2016].

[5] „PACE UNIVERSITY,“ Seidenberg school of computer, [WWW].

Available: http://csis.pace.edu/~marchese/DPS/Lect3/dpsl3.html. [01 05 2016].

[6] R. T. Azuma, „A Survey of Augmented Reality,“ *Teleportation and Virtual Environments*, 1997.

[7] M. Tikander, Artist, *Development and Evaluation of Augmented Reality Audio Systems.* [Art]. 2009.

[8] „Hear & There,“ Hear & There, [WWW].

Available: http://hear-there.com/. [12 05 2016].

[9] P. Y. Baram, „Introduction the Audio-Visual Walkert for Gait Improvements,“ 2007. [WWW].

Available: http://medigait.com/main/. [24 04 2016].

[10] „Geomagic,“ [WWW].

Available: http://www.geomagic.com/en/products-landing-pages/haptic. [24 04 2016].

[11] C. S. F. B. K. S. Dan Morris, „Visuohaptic Simulation of Bone Surgery for Training and Evaluation,“ *Virtual and Augmnented Reality Supported Simulators*, 2006.

[12] T. K. S. U. H. Y. Christian SANDOR, „Exploring Visuo-Haptic Mixed Reality,“ Institute of Electronics, Infor- mation and Communication Engineers, 2007.

[13] R. R. M. I. 12. Oliver Bimber, „Spatila Augmented Reality,“ *SIGGRAPH*, 2007.

[14] „Rift Info,“ [WWW].

Available: http://riftinfo.com/oculus-rift-history-how-it-all-started. [23 04 2016].

[15] „View Master Source,“ [WWW].

Available: http://www.vmresource.com/camera/cameras-general.htm. [16 04 2016].

[16] „xbox,“ Microsoft, [WWW].

Available: http://www.xbox.com/en-US/xbox-360/accessories/kinect. [23 04 2016].

[17] „hinnavaatlus,“ [WWW].

Available: http://www.hinnavaatlus.ee/products/M%C3%A4ngud+ja+konsoolid/Konsoolide+lisaseadmed/284459/. [23 04 2016].

[18] „xbox,“ Microsoft, [WWW].

Available: http://www.xbox.com/ru-RU/xbox-one/accessories/kinect-for-xbox-one#fbid=Ddlj5kcLv5n. [23 04 2016].

[19] „hinnavaatlus,“ [WWW].

Available: http://www.hinnavaatlus.ee/products/M%C3%A4ngud+ja+konsoolid/Konsoolide+lisas

eadmed/509569/. [23 04 2016].

[20] „Unity,“ [WWW].

Available: http://unity3d.com. [23 04 2016].

[21] „CryEngine,“ Crytek, [WWW].

Available: https://www.cryengine.com/. [23 04 2016].

[22] „Unreal Engine,“ Epic Games, [WWW].

Available: https://www.unrealengine.com/what-is-unreal-engine-4. [23 04 2016].

[23] „OpenCV,“ [WWW].

Available: http://opencv.org/. [23 04 2016].

[24] „Vuforia,“ [WWW].

Available: http://www.vuforia.com/. [23 04 2016].

[25] „AR Toolkit,“ [WWW].

Available: http://artoolkit.org/. [23 04 2016].

[26] „autodesk.com,“ Autodesk, [WWW].

Available: http://www.autodesk.ru/products/3ds-max/overview. [23 04 2016].

[27] „autodesk.com,“ Autodesk, [WWW].

Available: http://www.autodesk.ru/products/maya/overview. [23 04 2016].

[28] „blender.org,“ Blender, [WWW].

Available: https://www.blender.org/. [23 04 2016].

[29] „GitHub,“ [WWW].

Available: https://github.com/mitchemmc/UE4FirstPersonVRTemplate. [24 04 2016].

[30] „Unreal Engine AR,“ [WWW].

Available: http://www.unreal4ar.com/. [24 04 2016].

[31] M. Hizer, „Marker Detection for Augmented reality Applications,“ *Computer graphics and vision,* 2008.

[32] K. H. W. a. M. M. Y. C. Ying Kin Yu, „Pose Estimation for Augmented reality Applications Using Genetic Algorith,“ 2008.

[33] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Boston, 1989.

[34] T. Linderberg, „Features Detection with Automatica scale Selection,“ 1998.

[35] „Mathworks documentation,“ [WWW].

Available: http://www.mathworks.com/help/vision/feature_detection_category_image.png. [24 04 2016].

[36] „AR Toolkit Documentation,“ [WWW].

Available: http://www.artoolkit.org/documentation/doku.php?id=3_Marker_Training:marker_training. [24 04 2016].

[37] „Wiki Unreal Engine,“ Epic Games, [WWW].

Available: https://wiki.unrealengine.com/Integrating_OpenCV_Into_Unreal_Engine_4. [05 01 2016].

[38] „wiki.unrealengine.com,“ Epic Games, [WWW].

Available: https://wiki.unrealengine.com/Integrating_OpenCV_Into_Unreal_Engine_4. [24 04 2016].

[39] „docs.unrealengine.com,“ Epic Games, [WWW].

Available: https://docs.unrealengine.com/latest/INT/Engine/Rendering/Materials/MaterialInstances/index.html. [24 04 2016].

[40] „USC Institute of Creative Technologies,“ [WWW].

Available: http://projects.ict.usc.edu/mxr/faast/. [25 04 2016].

[41] „unrealengine.com,“ Epic Gamges, [WWW].

Available: https://www.unrealengine.com/marketplace/kinect-4-unreal-introduction. [27 04 2016].

[42] „dev.ovrvision.com,“ [WWW].

Available: http://dev.ovrvision.com/doc_en/index.php?startup_manual. [27 04 2016].

[43] „http://www.inteco.com.pl/,“ http://www.inteco.com.pl/, [WWW].

Available: http://www.inteco.com.pl/. [01 05 2016].

[44] Blog [WWW].

Available: http://2.bp.blogspot.com/-38Cek3IxmGg/Tc4tLNpG9oI/AAAAAAAAAc8/Dd89HVBSFUk/s1600/skeleton.jpg. [25 04 2016].

[45] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[46] „Indoors,“ [WWW].

Available: http://indoo.rs/the-slam-engine-is-here/. [24 04 2016].

[47] „Wik Unreal Engine,“ Epic Games, [WWW].

Available: https://wiki.unrealengine.com/Integrating_OpenCV_Into_Unreal_Engine_4. [24 04 2016].

APPENDICES

Appendix 1. C++ code for web camera integration to UE4

WebCamReader.cpp

```
#include "MyProject6.h"
#include "Engine.h"
#include "WebcamReader.h"
// Sets default values
AWebcamReader::AWebcamReader()
{
	// Set this actor to call Tick() every frame.
	PrimaryActorTick.bCanEverTick = true;
	CameraID = 0;
	isStreamOpen = false;
	VideoSize = FVector2D(640, 480);
	Threshold = 120;
	buffer.reserve(1920 * 1080 * 3);
#ifdef __clang__
#pragma clang diagnostic ignored "-Wmissing-braces"
#endif
	map <int, std::array<float, 3>> markersLocations = {
		{ 11111111, { { 120, 150, 0 } } },
		{ 1100110011, { { 170.5, 150, 0 } } },
	};
	std::array<double, 9> cameraMatrixBuf = {
8.7666720553412040e+002, 0., 3.1147149867982830e+002,
		0., 8.7084221678578592e+002, 2.2823097838101620e+002, 0.,
0., 1. };
	std::array<double, 8> cameraDistortionBuf = { 0, 0, 0, 0, 0 };
	int markerHalfSize = 10; //cm
	markersDetector = new MarkersDetector(&markersLocations,
cameraMatrixBuf, cameraDistortionBuf, markerHalfSize);
}
// Called when the game starts or when spawned
void AWebcamReader::BeginPlay()
{
	Super::BeginPlay();
	isStreamOpen = markersDetector->captureCamera(CameraID,
VideoSize.X, VideoSize.Y);
	if (isStreamOpen)
	{
		UpdateFrame();
		VideoSize = FVector2D(VideoSize.X, VideoSize.Y);
		VideoTexture = UTexture2D::CreateTransient(VideoSize.X,
VideoSize.Y);
		VideoTexture->UpdateResource();
		VideoUpdateTextureRegion = new FUpdateTextureRegion2D(0,
0, 0, 0, VideoSize.X, VideoSize.Y);
```

```
            markersDetector->threshold = Threshold;
            //Initialize data array
            Data.Init(FColor(0,    0,    0,    255),    VideoSize.X    *
VideoSize.Y);
      }
}
void AWebcamReader::UpdateTexture()
{
      if (isStreamOpen && buffer.size())
      {
            //Copy Mat data to Data array
            for (int y = 0; y < VideoSize.Y; y++)
            {
                  for (int x = 0; x < VideoSize.X; x++)
                  {
                        int i = x + (y * VideoSize.X);
                        Data[i].B = buffer[i * 3 + 0];
                        Data[i].G = buffer[i * 3 + 1];
                        Data[i].R = buffer[i * 3 + 2];
                  }
            }

            //Update texture 2D
            UpdateTextureRegions(VideoTexture,   (int32)0,   (uint32)3,
VideoUpdateTextureRegion,   (uint32)(4   *   VideoSize.X),   (uint32)4,
(uint8*)Data.GetData(), false);
      }
}
void AWebcamReader::UpdateTextureRegions(UTexture2D* Texture, int32
MipIndex, uint32 NumRegions, FUpdateTextureRegion2D* Regions, uint32
SrcPitch, uint32 SrcBpp, uint8* SrcData, bool bFreeData)
{
      if (Texture->Resource)
      {
            struct FUpdateTextureRegionsData
            {
                  FTexture2DResource* Texture2DResource;
                  int32 MipIndex;
                  uint32 NumRegions;
                  FUpdateTextureRegion2D* Regions;
                  uint32 SrcPitch;
                  uint32 SrcBpp;
                  uint8* SrcData;
            };

            FUpdateTextureRegionsData*      RegionData      =      new
FUpdateTextureRegionsData;
            RegionData->Texture2DResource                                =
(FTexture2DResource*)Texture->Resource;
            RegionData->MipIndex = MipIndex;
            RegionData->NumRegions = NumRegions;
            RegionData->Regions = Regions;
            RegionData->SrcPitch = SrcPitch;
            RegionData->SrcBpp = SrcBpp;
```

```
            RegionData->SrcData = SrcData;

            ENQUEUE_UNIQUE_RENDER_COMMAND_TWOPARAMETER(
                  UpdateTextureRegionsData,
                  FUpdateTextureRegionsData*, RegionData, RegionData,
                  bool, bFreeData, bFreeData,
                  {
                        for (uint32 RegionIndex = 0; RegionIndex <
RegionData->NumRegions; ++RegionIndex)
                        {
                              int32   CurrentFirstMip   =   RegionData-
>Texture2DResource->GetCurrentFirstMip();
                              if        (RegionData->MipIndex        >=
CurrentFirstMip)
                              {
                                    RHIUpdateTexture2D(
                                          RegionData->Texture2DResource-
>GetTexture2DRHI(),
                                          RegionData->MipIndex          -
CurrentFirstMip,
                                          RegionData-
>Regions[RegionIndex],
                                          RegionData->SrcPitch,
                                          RegionData->SrcData
                                          +                   RegionData-
>Regions[RegionIndex].SrcY * RegionData->SrcPitch
                                          +                   RegionData-
>Regions[RegionIndex].SrcX * RegionData->SrcBpp
                                          );
                              }
                        }
                        if (bFreeData)
                        {
                              FMemory::Free(RegionData->Regions);
                              FMemory::Free(RegionData->SrcData);
                        }
                        delete RegionData;
                  });
      }
}

// Called every frame
void AWebcamReader::Tick(float DeltaTime)
{
      Super::Tick(DeltaTime);
      if (isStreamOpen)
      {
            UpdateFrame();
            UpdateTexture();
            OnNextVideoFrame();
      }
}

void AWebcamReader::UpdateFrame()
```

```
{
	if (isStreamOpen)
	{
		std::array<float, 3> camLocation;
		std::array<float, 3> camRotation;

		markersDetector->update(buffer, camLocation, camRotation,
usedMarkers);
		cameraLocation.X = nearestEvenInt(camLocation[0]);
		cameraLocation.Y = nearestEvenInt(camLocation[1]);
		cameraLocation.Z = nearestEvenInt(camLocation[2]);
		cameraRotation.Roll = nearestEvenInt(camRotation[0] * 360
/ PI);
		cameraRotation.Pitch = nearestEvenInt(camRotation[1] *
360 / PI);
		cameraRotation.Yaw = nearestEvenInt(camRotation[2] * 360
/ PI);
		//GEngine->AddOnScreenDebugMessage(-1, 1.0f,
FColor::Green, FString::FromInt(usedMarkers));
	}
}

void AWebcamReader::OnNextVideoFrame_Implementation()
{
	// No default implementation
}
void AWebcamReader::EndPlay(const EEndPlayReason::Type
EndPlayReason)
{
	Super::EndPlay(EndPlayReason);
	markersDetector->releaseCamera();
	isStreamOpen = false;
}
int AWebcamReader::nearestEvenInt(float to)
{
	int r = round(to);
	return (r % 2 == 0) ? r : (r + 1);
}	markersDetector->releaseCamera();
	isStreamOpen = false;
}
int AWebcamReader::nearestEvenInt(float to)
{
	int r = round(to);
	return (r % 2 == 0) ? r : (r + 1);
}
```

WebCamReader.h

```
#pragma once
#include "GameFramework/Actor.h"
#include <array>
#include <map>
#include "WebcamReader.generated.h"
UCLASS()
class MYPROJECT6_API AWebcamReader : public AActor
```

```
{
	GENERATED_BODY()
public:
	// Sets default values for this actor's properties
	AWebcamReader();
	// Called when the game starts or when spawned
	virtual void BeginPlay() override;
	virtual void EndPlay(const EEndPlayReason::Type EndPlayReason)
override;
	// Called every frame
	virtual void Tick(float DeltaSeconds) override;
	// The device ID opened by the Video Stream
	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = Webcam)
		int32 CameraID;
	// If the stream has succesfully opened yet
	UPROPERTY(BlueprintReadWrite, Category = Webcam)
		bool isStreamOpen;
	// The videos width and height (width, height)
	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = Webcam)
		FVector2D VideoSize;
	UPROPERTY(BlueprintReadWrite, EditAnywhere, Category = Webcam)
		int32 Threshold;

	// The current video frame's corresponding texture
	UPROPERTY(BlueprintReadWrite, Category = Webcam)
		UTexture2D* VideoTexture;
	// The current data array
	UPROPERTY(BlueprintReadWrite, Category = Webcam)
		TArray<FColor> Data;
	// Blueprint Event called every time the video frame is updated
	UFUNCTION(BlueprintNativeEvent, Category = Webcam)
		void OnNextVideoFrame();
	UPROPERTY(BlueprintReadWrite, Category = Webcam)
		FVector cameraLocation;
	UPROPERTY(BlueprintReadWrite, Category = Webcam)
		FRotator cameraRotation;
	UPROPERTY(BlueprintReadWrite, Category = Webcam)
		int32 usedMarkers;
protected:
	void UpdateTextureRegions(UTexture2D* Texture, int32 MipIndex,
uint32 NumRegions, FUpdateTextureRegion2D* Regions, uint32 SrcPitch,
uint32 SrcBpp, uint8* SrcData, bool bFreeData);
	FUpdateTextureRegion2D* VideoUpdateTextureRegion;

	void UpdateFrame();
	void UpdateTexture();
	MarkersDetector* markersDetector;
	std::vector<uchar> buffer;
	int nearestEvenInt(float to);
};
```