Markus Liuska

Augmented Reality

Opinnäytetyö KESKI-POHJANMAAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Toukokuu 2012

Yksikkö Ylivieska	**Aika** Toukokuu 2012	**Tekijä/tekijät** Markus Liuska
Koulutusohjelma Tietotekniikka		
Työn nimi Augmented Reality		
Työn ohjaaja FM Joni Jämsä		**Sivumäärä** 39
Työelämäohjaaja FT Mika Luimula		

Työssä tutkittiin, mitä lisätty todellisuus tarkoittaa ja miten sitä voidaan hyödyntää, kun kehitetään ohjelmistoja. Työn tarkoituksena oli kehittää ohjelmisto, joka hyödyntää lisättyä todellisuutta siten, että se parantaa käyttäjäkokemusta.

Sovellus kehitettiin käyttämällä AndAR-ohjelmistokehystä, jonka toiminta perustuu ARToolKit-kirjastoon. ARToolKit on yksi suosituimmista kirjastoista lisätyn todellisuuden ohjelmistokehityksessä.

Työn toinen tarkoitus oli auttaa uutta kehittäjää, jolla ei ole edellistä kokemusta tämäntyyppisestä kehitystyöstä. Tämän vuoksi opinnäytetyö sisältää teoreettisen osan, joka kertoo, mitä lisätty todellisuus tarkoittaa ja esittää käytännön esimerkkejä lisätyn todellisuuden käytöstä. Opinnäytetyön käytännön osa esittää, kuinka ARToolKit-kirjastoa voidaan hyödyntää ja kuinka AndAR-ohjelmistokehystä käytettiin Smart Picture Studio -projektia varten.

Asiasanat
ARToolKit, Augmented Reality, Programming, Software development

CENTRAL OSTROBOTHNIA UNIVERSITY OF APPLIED SCIENCES	Date May 2012	Author Markus Liuska
Degree programme Information Technology		
Name of thesis Augmented Reality		
Instructor M.Sc. Joni Jämsä		Pages 39
Supervisor Ph.D. Mika Luimula		

In this thesis, research was made on what Augmented Reality means and how it can be utilized when developing software applications. The main purpose of this thesis was to create an application that uses Augmented Reality to improve the user experience.

The application was created by using AndAR which is a framework based on the ARToolKit. The ARToolKit is nowadays one of the most popular libraries for creating Augmented Reality applications.

The secondary purpose of this thesis was to help the new developer who has no background on Augmented Reality or its practical applications. Therefore the thesis includes a theoretical part that explains what Augmented Reality is and describes real life implementations and use case scenarios. The practical part of the thesis shows how the ARToolKit library can be utilized and how AndAR was used for Smart Picture Studio project.

Key words
ARToolKit, Augmented Reality, Programming, Software development

Terms and Abbreviations

SNS	Social Network Service, online service that focuses on social relations among people
AR-Marker	Augmented Reality marker, what is used for rendering 3D-object
Library	Collection of different resources, used for developing software
UML	Unified Modeling Language, general-purpose modeling language
GPL license	GNU General Public License, copyleft license for general use
HMD	Head-mounted display

TABLE OF CONTENTS

1 INTRODUCTION

In the past few years Augmented Reality applications are starting to appear for average users. The main reason for this wave of new applications is the development in mobile computing. While restrictions from technical side are getting lesser, more opportunities for implementing novel ideas into reality have emerged. While the hardware aspect for Augmented Reality is crucial for future, the human-computer interactions are as important for the future development. In this field of research and development, new ways of using technology are implemented together with Augmented Reality.

Many researches are stating that Augmented Reality can be a highly important technology in the future. The current state of technology is not allowing for the futuristic usage that is usually presented in science fiction. The idealistic augmentation is not so far as the public thinks. This is mostly caused by unreleased research studies, which are done in the commercial field.

The main purpose of this thesis is to create a base of information for the new developer who has no previous experience in this field. Because the nature of Augmented Reality as a technology thesis consists of multiple examples of different types of prototypes created in past and practical applications that are used in different industrial fields as well as information about the definition of the Augmented Reality term.

As stated earlier, the thesis is intended for the new developer. For this reason, the end of the theoretical part is presenting a tool for Augmented Reality development. This tool for software development is a library called ARToolKit, which is used in many different projects worldwide. The main reason for choosing this tool is the practical side of this thesis, which is using development library which is based on ARToolKit.

2 AUGMENTED REALITY

The main purpose of this chapter is to explain what Augmented Reality means and what it can be used for. Because of the wide area of this technology, information is given in the form of examples and graphs that are presenting different ways for using Augmented Reality.

2.1 Definition

The definition of *Augmented Reality* (AR), in this thesis is using the same definition as Ronald T. Azuma. There AR is presented as a variation of Virtual Environments (VE). (Azuma, 1997, 2.)
In VE, also called Virtual Environment, the user usually needs to stay in a certain area. In contrast AR allows user to interact with the real world, without limiting the mobility of the user. This means that AR is supplementing reality, instead of completely replacing it. In the ideal situation, the user and virtual objects would appear to be in same space.

In a paper ”A survey of Augmented Reality” Azuma gives an example of a fictional situation, where virtual objects and the real world are combined. This was created by special effects in the film Who Framed Roger Rabbit. Another good example of ideal usage of AR could be science fiction from Japan, called Denno Coil. In this animation, people have glasses that give possibility of seeing virtual objects while connected into city wide network.

For avoiding the limitation of AR to specific technologies, Azuma's survey defines AR something that:
1. Combines real and virtual
2. Is interactive in real time
3. Is registered in 3-D

R. Azuma writes in his paper ”This definition allows other technologies besides Head-Mounted-Displays (HDMs) while retaining the essential components of AR” (Azuma, 1997, 2.).

2.2 Possibilities of augmentation

By following the definition from the beginning of this thesis, we can find that AR is not only limited to certain usage with HMD. This leaves the road open into many possible directions (Azuma 1997, 9.)

People understand the reality trough all senses. This means that there are many other possibilities with AR than just human sight. Most of the progress in this field is done by creating something with visualization. But recently there has been progress with other senses.

Smell and taste

In year 2010 Tokyo University has published research with the experimental title Meta Cookie. This experiment is created by using HMD and ARToolKit. Because of ARToolKit's nature what demands a visual AR marker, shape for recognition is created by branding the cookies with hot iron.

The effect would not be completed by just cheating the brain by false images. Because of this, researchers have installed a device under HDM, which creates a correct smell. This alone is enough to create an illusion of different tastes. (Narumi et al. 2010.)

Hearing

There have been many studies about Augmented Reality Audio (ARA). An example of development in this area is a project called NAVIG. Being developed in France it is meant for blind people. The system is using a Head-Mounted-Camera HMC, head phones, microphone and computer. The user can control the system through voice commands. He can find physical items that are recognized by computer vision and 3D audio is informing user the correct direction of the object.

Naturally in the current state of the project is too early for actual usage with blind people. Still, this is providing a good example where augmentation is used for more than entertainment usage. (NAVIG 2012.)

Touch

Japanese company called NTT COMWARE has created a glove that allows the user to feel different 3D objects. (Pink tentacle 2012.) For example, the system allows a “virtual handshake” over the network. This gives many interesting possibilities for combining the virtual into reality. For example by combining this technology with HMD, the feeling of being inside a virtual world is largely enforced.

While usage of the physical glove is limiting the user, there are different approaches for creating an illusion of touching a virtual object. For example the University of Tokyo has released Touchable Holography in SIGGRAPH event in year 2009. Their system was using a holographic display that is provided images from LCD by utilizing a concave mirror, the image appearing to be floating in the air.

Actual illusion is of feeling the virtual object is created by using ultrasound, what is only effective at focal point. The focal point is changing depending of hand movements, what is tracked by Infrared sensors. This gives new possibilities by allowing the user to touch virtual objects without any restriction of movement. (Hoshi et al. 2009.)

2.3 Brief History

Technically the history of augmented reality starts from the moment, where the term was created. Naturally there has been much of research even before the definition existed. The purpose of this chapter is not to show every project that has been developed in the past, but instead showing some important steps that were crucial for the future development of AR.

NaviCam 1995

NaviCam (NAVIgation CAMera) was created by Jun Rekimoto and Katashi Nagao. The system used quite a powerful workstation what had a camera mounted on the mobile screen. The system was detecting code-colored markers by processing live feed from camera. When a marker was detected, the system displayed context sensitive information on the top of the video feed. (Rekimoto & Nagao, 1995.)

Touring Machine 1997

Touring Machine (MARS) is the first mobile Augmented Reality system. It was using HMD for showing the information and backpack which contained a computer, GPS and digital radio for web access. This system was one of the first that allowed a person to actually move around different areas, without restrictions created by laboratory environment. (Feiner et al. 1997.)

AR-Quake 2000

AR-Quake was presented by Bruce Thomas. The idea was based on one of the most popular FPS games in history. The system used vision-based tracking, GPS and a digital compass for trying to create an illusion of player being inside a game. The computer was wearable by using a backpack and interaction with game was handled with a simple device with two buttons (Thomas et al. 2000).

Indoor guidance system 2003

It was one of the first AR applications that was running on average consumer device. One could say that it is a beginning of the current flow of AR applications in smart phones. The application was running autonomously on a PDA and it used Windows mobile port of ARToolKit. The main purpose of this system, besides of being a prototype of the technology was to show information for the user about different locations inside a building (Wagner & Schmalstieg 2003).

2.4 Practical usage of AR

Most of the actual applications centering on AR are created for demonstrations or research purposes. There are still actual practical applications what are used. For a type of technology that is still under heavy development, any kind of practical implementations are crucially important. This raises the general interest for the technology and makes commercial potential possible.

2.4.1 Military

It is not unusual to see first real practical applications for new technology used in military purposes. This does not change with AR technology either. Naturally most of the military projects are not open for public. Still it has been public information for a long time, that fighter pilots are using AR as the guidance system. (Azuma 1997, 9.)

ARMAR

Augmented Reality for Maintenance and Repair (ARMAR) developed under Columbia University's Computer Graphics & User Interfaces Lab. The system presents visual information for the user about steps what are needed for maintaining or repairing purposes.

The visual information is provided for the user by using head mounted display, this allows free hands for doing the needed steps by following the instructions from the system. (Henderson & Feiner, 2009.)

2.4.2 Medical

Currently there are many AR implementations developed for the medical usage. Because the current state of the technology, most of these applications are concentrated on training medical workers.

CAE ProMIS

This system owned by CAE, is developed for training doctors. Main purpose of the system is to give the possibility to train realistic surgery situations, without risking actual patients in the middle of process.

The system has multiple different modules, which can be used for changing the situation. This allows the training to be more dynamic and decreases the costs of having a specialized simulator for every situation. (CAE 2012.)

2.4.3 Commercial

Lately many big companies have started to use augmented reality as a tool for advertising. This is usually done by using customer's mobile device, but there are cases where the AR system is built in a certain location, for advertising purposes. In this chapter, some of these implementations are presented.

Starbucks

While ago the Starbucks launched an advertising campaign, which included an application that presented animation for the user. Instead of plain markers, the system used images that were printed, for example on paper cups or advertising on the wall. After the user has found the item, he can watch the animation and share the results with friends. (Starbucks 2012.)

Lego

Toy company Lego has released a system that has been placed in retail stores. When the user has a package of Legos and presents it for the system, the virtual already built structure appears on top of the package for informing the user about the inside. (Notcot 2012.)

Amazon

Amazon is currently offering an application for mobile device that provides information about books and games. The system works by following way: The user scans cover of the book by using a mobile device's camera. After the cover is scanned, the results are compared against Amazons huge database. If the item was found, results are presented for the user. (A9 2012.)

2.5 Future

Even though the AR technology has been under research for many years, it is still hard to predict, how the technology will affect an average person's everyday life. Still there are few projects that give glimpses of possibilities in future.

AR contact lenses

Generally the approaches are using either the mobile devices display, or HDM. Under development there is an interesting technology that is creating more possibilities of augmentation without forcing the user to wear a device. This is done by creating a small circuit in the contact lenses. The University of Washington has done research that provided a prototype. This prototype is currently allowing only presentation of a single pixel, created by led. For ensuring the safety of the contact lenses, they have done testing with animals, more specifically rabbits.

Researchers of Washington University were ensuring that rabbits were not harmed by the usage of contact lenses. Current technology allows only one pixel to being presented. There are already possibilities for creating applications over this technology. For example the deaf user can be warned about danger and because the nature of this type of information. It is difficult for a user not to notice it. (Spectrium IEEE 2012.)

2.6 Ideal Augmentation

Within his definition of Augmented Reality, Azuma had written about ideal situations, where the user and virtual objects could coexist on the same reality. (Azuma 1997, 2.) Naturally this is still science fiction, but it should be the aim for this type of applications.

For reaching this hard goal, the technology should continue evolving in same way as it has until now. The biggest issue is the hardware that is required for this type of usage. Average persons are not willing to wear required hardware, if the hardware is creating unwanted attention or the usage of the device is not practical enough.

Needed infrastructure

In the most ideal situation, we could apply augmentation in real time anywhere in the world. The problem is to have united location for data, for example. Different companies are releasing their own solutions for the public. The technology could

be limited by the competition between companies. Naturally creating the idealistic world, where virtual world would co-exist within reality is an unrealistic ideal. Even while being unrealistic, there is still possibility for this in future.

2.7 Hardware

Currently there are many rumors about how big companies are trying to find a solution for displaying information to the user. This solution can be some type of goggles or contact lenses. The breakthrough has not still happened. There are still some companies that are starting to step ahead with presenting their results. For example the company Vuzix has already released its second version of see-trough display goggles. (Vuzix 2012.)

Hardware is still too expensive for normal users, but if the technology is advancing the prices might drop a lot in a few years. As stated earlier, there is the probability that an average person would not walk in the streets while wearing this type of glasses because they are not blending well with the other clothing. The biggest problems with current technologies are following: Size, Price and power.

While the parts of hardware are getting smaller, the need of power is still creating big difficulties for the devices. Even in case where normal looking see-trough displays could be produced. The need of loading batteries can cause some serious difficulties for using the system. This said, the future of this type of ideal augmentation is depending a lot for the development of cameras, processors and another types of components.

Vuzix STAR

A company called Vuzix has released already few goggles for AR use specially. These goggles are allowing the user to see the virtual objects, but still by using their see-trough technology for allowing the user to experience the real world at the same time. (Vuzix 2012.)

Golden-i

While ago company called Motorola has released information about a prototype of a headset. This headset allows user's hands to be free, by using voice commands. The headset is designed for fitting under a construction helmet and while the user is wearing the headset. The display imitates 15-inch display. (MyGoldenI 2012.)

3 ARTOOLIKIT

ARToolKit is a software library, written in C and C++ languages. It was developed in 1999 by M. Billinghurst and H. Kato. And it is still under a continuous development. The open source library is released under GNU (General Public License) and it is used around the world for personal and academic projects. (ARToolKit 2012.)

3.1 AR markers

It is not unusual for AR applications to use physical markers for adding virtual objects into real world. There are commercial tools that can create markers from normal everyday images. (Qualcomm 2012.) This brings much more possibilities when it comes to creating a system that is centered on AR.

Generally projects that are using ARToolKit are using simple black and white markers. It is possible to generate markers, from a normal image data. But this method might add risk of failure at marker detection.
The application that is implementing ARToolKit, needs also data which is used for comparisons with physical object. The actual data that is used is in human readable format, and it's basically a simple array of values. (ARToolKit 2012.)

GRAPH 1: Classic Hiro pattern of marker, used in many project

Creation of AR markers

There are two ways to create a marker. Generating data and printing the shape by using this information or a more common way, which is to print the marker and generate the data by using image or video. (ARToolKit 2012; Flash Tarotaro 2012.)

The quality of the image is really important. It is possible that data generated from picture is causing troubles even if the marker is not visible. This is caused by the way which ARToolKit is using for recognition. In case of a blurry image, the application might recognize the marker even from a white wall.

3.2 Internal working of ARToolKit

Naturally the pattern recognition uses quite complex algorithm, but for basic understanding of the internal workings of ARToolKit, it is better to simplify the whole process. In this case, the whole process is simplified as six steps. (ARToolKit 2012.)

1. The camera is capturing video of real world and it's received frame by frame by ARToolKit.

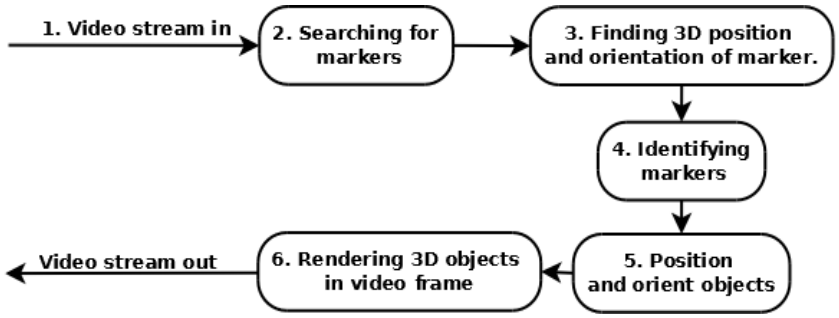
2. When a frame arrives it's converted into a binary image and scanned for any square patterns.

3. In case a square has been found, calculations for getting position of the camera relative to the square are made.

4. After finding the square and the calculations have been made, comparisons between symbol of captured marker and markers from memory are made.

5. When symbol is found from the memory, virtual objects are aligned with markers by using 3D transform.

6. The 3D object is rendered into the correct position within the video frame.



GRAPH 2: The float chart of marker recognition (adapted from ARToolKit, 2012)

3.3 Limitations

Like any another vision based recognition, ARToolKit has some limitations that are necessary to notice. The main limitations are following: range, pattern complexity, orientation and lighting conditions. It is obvious that if a human cannot recognize a pattern from a really shallow angle, it is almost impossible for the algorithm. (ARToolKit 2012; Kato & Billinghurst 1999.)

Distance

The following table from official documentation of ARToolKit, presents the possible distances and pattern sizes for AR marker.

TABLE 1: Range limitations (adapted from ARToolKit, 2012a)

Pattern size (Inches)	Usable Range (Inches)
2.75	16
3.50	25
4.25	34
7.37	50

Distance itself can cause problems with pattern recognition, but there is also another problem that might be caused by the nature of ARToolKit's internal working, what is related to distance between the camera and AR marker. If the actual physical size of AR marker is not matching with the information what is read from generated file. The information about distance between camera and AR marker is false.

Lighting and reflection

The quality of the hardware is crucial for pattern recognition. The chances for failing recognition are rapidly increasing when the hardware cannot accomplish the requirement of ARToolKit. Naturally this should be always considered when choosing hardware.

The largest amount of failures could be caused by lighting, if the environment is too dark, or in some cases too bright. It should be noted, that even in optimal lighting, the material of AR marker can cause problem situations for example if the material is highly reflective. (ARToolKit 2012; Kato & Billinghurst 1999.)

Shapes and pattern recognition

Optimally ARToolKit only recognizes AR markers, but this works only in the environment where the shapes are limited. Naturally in real life environment this is not possible. This means that depending on amount of different registered AR markers, the amount of false recognitions are increasing. (ARToolKit 2012; Kato & Billinghurst 1999.)

3.4 Tools based on ARToolKit

While ARToolKit is used in many different projects, there are some situations where using the original version of ARToolKit is unpractical, for example. Google's Android operating system is using Java as main programming language. For this type of cases there might be ready made version for needed operating system.

TABLE 2. Different tools based on ARToolKit

Name	Platform	Language
ARToolKitPlus	*	C++
NyARToolKit	*	C++, C#, Java, AS3
osgART	*	C++
FLARToolKit	*	AS3
ArUco	*	C++
AndAR	Android	Java
SLARToolKit	WP7	Silverlight

3.4.1 NyARToolkit

While many projects are using ARToolKit, it's spinnoff NyARToolKit is gaining more popularity. It is released under GPLv3 License but some modules might include MIT license. It has support for many different programming languages, including C++, C#, AS3 and Java. It should be noticed, that when this thesis were being written, the C++ version of NyARToolkit was still missing major functionality. (NyARToolKit 2012.)

3.4.2 SLARToolKit

SLARToolKit is a library, used with Silverlight and Windows phone. The aim of the open source project is to make an easy way for create applications with Augmented Reality. While SLARToolKit does have a base in ARToolKit, it is actually based on earlier presented NyARToolKit. The library is released under dual license, which means that it can be used as open source project and in closed projects. The open source license is based on General Public License. (SLARToolKit 2012.)

3.4.3 osgART

osgART is cross-platform project, that has main purpose of simplifying the development of Augmented reality applications. It uses multiple different libraries as base. The main programming language for this library is C++ language, which makes porting the application more simple into different platforms. The library is developed by Human Interface Laboratory New Zealand (HIT Lab NZ).

The functionality includes high level integration of video input, spatial registration and photometric registration. Currently it has official support for Python, Lua and Ruby scripting languages. (osgART 2012.)

3.4.4 ARToolKitPlus

As it is stated in the documentation ARToolKitPlus is trying to continue the work of original ARToolKit. Even if the main library is still under active development, the

ARToolKitPlus is gives an advanced approach for using C++ programming language. This reduces the amount of work. For example, there is no need to create C++ wrapper for the ARToolKit's library functions.

It should be noted, that the library is at for more experienced C++ programmers. The library is released under GPL license. (ARToolKitPlus 2012.)

3.4.5 FLARToolKit

FLARToolKit is AS3 ported version from ARToolKit. The AS3 is scripting language that is used in Flash applications. The project has helper classes for current major 3D engines, including: Papervision3D, Away3D, Sandy and Alternativa3D.

The library is not actually ported straight from the ARToolKit. Instead it uses the same tactic as SLARToolKit, which is ported from NyARToolKit. It should be noted that usage of this library can give good advance against native applications, because the FLARToolKit is allowing easily spread applications, without forcing users download any extra applications. (FLARToolKit 2012.)

3.4.6 AndAR

AndAR is created for enabling easy development on Android platform. The library is open source project, which is licensed by using General Public License. At the current state of the project, the 3D development is done by using OpenGL. Language of choice in this project is the Java programming language. This creates more possibilities, because Java is the main supported language for the platform. (AndAR 2012.)

3.4.7 ArUco

ArUco is library what is based on ARToolKit and OpenCV projects. The programming language of choice in this project is C++ programming language. The main purpose of this project is following the same mentality as previously presented libraries, which is to simplify the process of creating AR application.

The 3D rendering can be implemented by using OpenGL or Ogre3D. (ArUco 2012.)

3.5 Development with ARToolKit

The main purpose of this chapter is to inform reader about how development can be started with ARToolKit. The chapter will go through the steps for compiling ARToolKit library and presents analysis of a simple example, which is provided within ARToolKit's development files.

3.5.1 Compiling ARToolKit library

In this chapter the steps of compiling process are explained. It should be noted that these instructions are made for Linux operating system, more specifically the Ubuntu distribution. If compiling needs to be on another platform, instructions about this can be found from ARToolKit's project website. (ARToolKit 2012a.)

Installing required packages

For compiling the library, following packages for development were installed.

- freeglut3-dev (For some OpenGL functions and creating a window)
- libv4l, (general operations related to video operations)
- libxi-dev (for allowing input from user)
- libxmu-dev (for easier usage of X libraries)
- libgstreamer0.10-dev (for video processing)

It should be noted that different versions of Ubuntu distribution or previously installed applications might cause dependency problems within packages. Information about possible missing libraries will be found while compiling the actual library.

For installing the packages, tool called Advanced Packing Tool (APT), was used. APT tool automates whole process of downloading and installing required packages. (APT 2012.)

The following commands were written on Ubuntu distributions GNOME Terminal.

```
apt-get install  freeglut3-dev libv4l-dev libxi-dev libxmu-dev
apt-get install libgstreamer0.10-dev
```

It should be noted, that apt-get command requires administrators rights for installing new packages on the operating system.

Downloading and compiling ARToolKit library

For compiling the ARToolKit library, latest usable archive of the project was downloaded from repository, what was located at SourceForge's server. Following commands were made for unpacking the archive and opening directory.

```
tar -xzvf ARToolKit-2.72.1.tgz
cd ARToolKit/
```

First step for compiling the actual code is to run script for configuring, named simply as Configure.

```
./Configure
```

After starting the script, it offers following list for the user.

```
Select a video capture driver.
  1: Video4Linux
  2: Video4Linux+JPEG Decompression (EyeToy)
  3: Digital Video Camcoder through IEEE 1394 (DV Format)
  4: Digital Video Camera through IEEE 1394 (VGA NCOMP Image Format)
  5: GStreamer Media Framework
```

It should be noted, that depending of different packages installed on operating system, the list might vary in different ways. In this case the selected driver for video capture was number 5. GStreamer, that was installed at earlier point. (GStreamer 2012.)

After running the script for configuring, compiling process can be started by calling make command from the same directory.

```
make all
```

If all the required packages were installed on the system, binaries of all the examples should be found from binary folder, which is located at root of the folder.

In cases of possible missing packages or other type of problems, it can be helpful to clean the project after this type of problems. This can be done by calling following command.

```
make clean
```

3.5.2 Simple example

Previous chapter presented how the ARToolKit was compiled. It also mentioned about how the examples were compiled into binary folder. In this chapter one of those examples is analyzed for better understanding of ARToolKit.
It should be noted that code snippets, which are presented in this chapter might not match fully with original source code (SimpleTest.c). Some lines are modified for easier presentation.

Presentation of functions
Following table presents shortly the purposes of different functions, declared at SimpleTest.c source file. For saving space the parameters or types are not included in the table.

TABLE 3. Functions of SimpleTest application

Name of function	Purpose of function
void init()	Opens video source with parameters, loads AR marker's data file and creates a window.
cleanup()	Calls cleanup function after stopping video capture.
keyEvent()	Waits for user to press ESC, for stopping application.
mainLoop()	Grabs video frame and tries to detect AR marker.
draw()	Uses OpenGL for drawing the cube by using Glut.

Presentation of global variables
All types that are presented in type column in following table are the base types, except the cparam, what is type of ARParam struct, defined in ARToolKit's documentation. (ARToolKit 2012b.)

TABLE 4. Global variables defined in the beginning of SimpleTest source

Name of variable	Type	Purpose of variable
xsize, ysize	int	Length and width of captured image.
thresh	int	Holds threshold value, used for creating a binary image.
count	int	Frame counter.
*cparam_name	char	Camera's settings stored in file.
cparam	ARParam	Intrinsic parameters of camera.
patt_name	char*	File path into AR marker's pattern data.
patt_id	int	Id of AR marker's pattern data.
patt_width	double	AR marker's pattern width.
patt_center	double[2]	Coordinates for AR marker's physical center.
patt_trans	double[3][4]	The transformation matrix. (Relative position of real camera to the real marker)

Main function

The purpose of main function is to run all initializing processes and then start actual video capturing.

In the beginning glutInit is called with parameters, which are giving possibility to change settings within window system. (Kilgrad, 2012, 2.)

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    init();
```

In the beginning glutInit is called with parameters that are allowing user to change wanted settings. The init function is called for initializing ARToolKit. Internal working of this function is presented later.

```
    arVideoCapStart();
    argMainLoop( NULL, keyEvent, mainLoop );
    return (0);
}
```

A separate thread for capturing video is started by calling arVideoCapStart function. After video capture has started, argMainLoop function is taking pointers to keyEvent and mainLoop functions.
This function will run, until the application is closed. It should be noted, that keyEvent and mainLoop functions are not called directly, instead the ARToolKit is handling these calls behind the scenes. (ARToolKit 2012b.)

Function for key events

The purpose of this function is to take possible key presses from the user and check if ESC-key has been pressed. This is done by comparing value of the key with hex value 0x1b (ESC-button's code). If the key code matches, cleanup function is called and exit is called for closing the application. (ARToolKit 2012b.)

```
static void   keyEvent( unsigned char key, int x, int y)
{
    if( key == 0x1b ) {
        printf("*** %f (frame/sec)\n", (double)count/arUtilTimer());
        cleanup();
        exit(0);
    }
}
```

Main loop function

Main loop function is responsible of finding the AR markers and rendering 3D object on wanted location. For being able to use main loop function with ARToolKit, it is required that the type of function and parameter is set as void. (ARToolKit 2012b.)

```
static void mainLoop(void)
{
```

At first, the needed variables are declared. Pointer called dataPtr is responsible of holding the actual image which is captured from video source and ARMarkerInfo type of struct, holds the information about detected marker.

```
    ARUint8         *dataPtr;
    ARMarkerInfo    *marker_info;
    int             marker_num;
    int             j, k;
```

Video frame is captured by using arVideoGetImage function. This data is used by dataPtr pointer what has the latest information available. If function is returning

NULL pointer, the data was not currently available. In this case arUtilSleep function will make thread waiting for 2 milliseconds and escaping from function by using return statement. (ARToolKit 2012b.)

```
if( (dataPtr = (ARUint8 *)arVideoGetImage()) == NULL ) {
    arUtilSleep(2);
    return;
}
```

Variable named count is responsible for keeping track of frames. If the value is set as zero, arUtilTimerReset function is called for resetting ARToolKit's internal timer. After this step, the rendering context is set as 2D mode by calling argDraw2D function and argDispImage function is called for rendering the captured video frame by giving dataPtr pointer.

```
if( count == 0 ) arUtilTimerReset();
count++;

argDrawMode2D();
argDispImage( dataPtr, 0,0 );
```

Marker detecting is done by arDetectMarker function. The function analyzes the captured video frame which is given in dataPtr pointer. It is converted into binary image format and searched for recognizable patterns. In case of error, cleanup function is called and application is closed. If calling was successful, function arVideoCapNext is called for grabbing the next video frame. (ARToolKit 2012b.)

```
if( arDetectMarker(dataPtr, thresh, &marker_info, &marker_num) < 0 ){
    cleanup();
    exit(0);
}
arVideoCapNext();
```

Checking the visibility of AR marker is left for following for a loop. This loop is going through the information that is received from arDetectMarker function. If marker was not detected, argSwapBuffers function and return statement will be called.

```
k = -1;
    for( j = 0; j < marker_num; j++ ) {
        if( patt_id == marker_info[j].id ) {
            if( k == -1 ) k = j;
            else if( marker_info[k].cf < marker_info[j].cf ) k = j;
        } }
```

```
    if( k == -1 ) {
        argSwapBuffers();
        return; }
```

For rendering the 3D object on AR marker the transformation between real camera and detected marker needs to be calculated. This is done by using arGetTransMat function. After the calculation, draw function is called and back buffer is switched with front buffer by using argSwapBuffers function. (ARToolKit 2012b.)

```
    arGetTransMat(&marker_info[k], patt_center, patt_width, patt_trans);
    draw();
    argSwapBuffers();
}
```

Initializing function

The init function is used for implementing settings for ARToolKit. In the beginning of the function ARParam type of struct is defined. This struct is responsible of holding camera intrinsic parameters. (ARToolKit 2012b.)

```
static void init( void )
{
    ARParam  wparam;
```

Firstly the video is opened by using given video configuration, after this the size of video image in pixels is saved into xsize and ysize variables.

```
    if( arVideoOpen( vconf ) < 0 ) exit(0);
    /* find the size of the window */
    if( arVideoInqSize(&xsize, &ysize) < 0 ) exit(0);
    printf("Image size (x,y) = (%d,%d)\n", xsize, ysize);
```

The intrinsic camera parameters are loaded by using arParamLoad function. If the function is returned successfully, cameras parameters are initialized after changing the size.

```
    if( arParamLoad(cparam_name, 1, &wparam) < 0 ) {
        printf("Camera parameter load error !!\n");
        exit(0);
    }
    arParamChangeSize( &wparam, xsize, ysize, &cparam );
    arInitCparam( &cparam );
```

Pattern is loaded by using arLoadPatt function. In case of error, the application is closing down with error message.

```
    if( (patt_id=arLoadPatt(patt_name)) < 0 ) {
        printf("pattern load error !!\n");
        exit(0);
    }
```

After another steps of initializing are made, the gsub library is initialized with parameters of camera.

```
    argInit( &cparam, 1.0, 0, 0, 0, 0 );
```

Function for cleaning up

Before application is closed it is cleaned by calling cleanup function. At first, the video capturing routine is stopped by calling function named arVideoCapStop. After this is done, video source is closed and argCleanup is called for closing gsub library. (ARToolKit 2012b.)

```
static void cleanup(void)
{
    arVideoCapStop();
    arVideoClose();
    argCleanup();
}
```

Drawing the 3D object

The most visible part of the application is naturally 3D rendering. This is made by function named draw. It should be noted, that the example uses old style of using OpenGL and it should not be used more than an example.

In the beginning some settings for lighting and reflection are made in form of arrays. These are crucial for making 3D object visible for the user.

```
static void draw( void )
{
    double    gl_para[16];
    GLfloat   mat_ambient[]     = {0.0, 0.0, 1.0, 1.0};
    GLfloat   mat_flash[]       = {0.0, 0.0, 1.0, 1.0};
    GLfloat   mat_flash_shiny[] = {50.0};
    GLfloat   light_position[]  = {100.0,-200.0,200.0,0.0};
    GLfloat   ambi[]            = {0.1, 0.1, 0.1, 0.1};
    GLfloat   lightZeroColor[]  = {0.9, 0.9, 0.9, 0.1};
```

For being able to render 3D objects on the context, argDrawMode3D is called. Following function call, argDraw3dCamera is called for updating camera parameters by doing this, it is complementing earlier call. After this clear depth buffer for openGL and some flags are set. (OpenGL 2012b.)

```
    argDrawMode3D();
    argDraw3dCamera( 0, 0 );
    glClearDepth( 1.0 );
    glClear(GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LEQUAL);
```

For transforming patt_trans array into matrix format what OpenGL supports, function named argConvGLpara function is called. The results are returned in format what is compatible with OpenGL's GL_MODELVIEW matrix mode. (OpenGL 2012b.)

```
    argConvGlpara(patt_trans, gl_para);
    glMatrixMode(GL_MODELVIEW);
    glLoadMatrixd( gl_para );
```

As stated earlier, lighting important for making 3D object visible for the users, this is made by using following OpenGL's functions.

```
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightfv(GL_LIGHT0, GL_AMBIENT, ambi);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightZeroColor);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_flash);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_flash_shiny);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMatrixMode(GL_MODELVIEW);
```

In the end the cube which is going to be rendered is moved by 25 units in Z axis and drawn by using glut's function glutSolidCube with size of 50 units. (Kilgrad 2012, 36-37.)

```
    glTranslatef( 0.0, 0.0, 25.0 );
    glutSolidCube(50.0);
    glDisable( GL_LIGHTING );
    glDisable( GL_DEPTH_TEST );
}
```

4 SMART PICTURE STUDIO

Smart Picture Studio (SPS) is a prototype application that is developed for Android platform. The main purpose of this application is to automate the process of taking photos from small items and selling those items in certain social networks.

This is done by placing the AR markers around physical object, which the user wants to sell. The application recognizes the AR markers around the object and uses this information for choosing the social network automatically. The type of physical object can be also described by adding special marker that sets the item type.

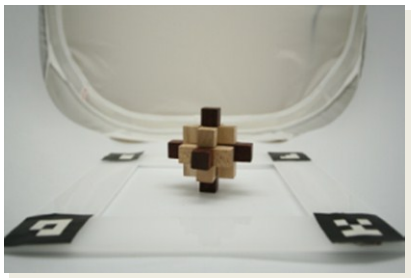
GRAPH 3. SPS application in use

4.1 Specifications

When the development of Smart Picture Studio was started, the following specifications were made for the project.

- Application is for Android platform
- Supported AR markers for social networks and item types
- AR markers for later calculations

It should be noted, that this chapter does not present the specifications which are not AR related.



GRAPH 4. Macro photo studio

4.2 Development tools

Naturally the specifications that were presented in previous chapter, affect the choices which were made before development. In this chapter the chosen development tools are presented shortly.

Used hardware

Because the specifications that were stated previously, the first factor for the choice was the required Android platform.
Another factor what had huge role, while choosing the proper device, was the need of high performance.
After research the Samsung's Galaxy Nexus was chosen. The device has enough power for pattern recognition and rendering the 3D models, which is crucial for running AR application. (Samsung 2012.)

Programming Language

The chosen programming language for this project was Java. The main reason for this is the fact that it is the officially supported language for Android platform and the amount of material and documentation. (Android, 2012c.)

Development environment

When Integrated Development Environment (IDE) was chosen, the support for the Java programming language was the most important factor. Naturally the need for good base for Android development was also crucial.
The chosen IDE for this project was NetBeans, that was found to be easy to install and use. It should be noted, that before using NetBeans Eclipse was tested, but it was too unstable. (NetBeans 2012; Eclipse 2012.)

Android Development with NetBeans

It should be noted, that while the IDE named Eclipse is officially supported by Google, the owner of Android operating system, development of Android applications is still possible by using other IDE's. In case of NetBeans, the development needs third a party plugin named NBAndroid. This plugin is an open source and it has been released under Apache 2.0 license. (NBAndroid 2012.)

AR Development tool

While following the original specifications and the chosen programming language, the usage of original ARToolKit was rejected. This was caused by the fact, that ARToolKit uses C programming language.

It should be noted that the development of C programming language is possible on Android platform. (Android 2012a.) The decision was made because the time limitations that the project had.
After the research of different tools what were based on ARToolKit. The two following libraries had better qualities for the project and were tested by using the example applications.

- NyARToolKit
- AndAR

After testing the example applications which were shared with the libraries, the chosen library was AndAR. Reasons for this choice were the test performance and the more clear documentation. It should be noted that the performance testing was made purely by human eye. While the difference with performance was clear, the results not always were objective.

4.3 Marker types

At the current state of the application, the SPS application is supporting the marker types which are presented on this chapter. In the graph below, all the current marker types are rendered.

GRAPH 5. All marker types rendered with SPS application

Social network markers

Social network markers are created for choosing the wanted network, in which possible photos are going to be uploaded. The current version of the project is supporting following types of markers: Flickr, Ebay, Twitter and Facebook.

It should be noted, that only one of the social network markers is supported when taking a picture. This is caused by logic for uploading the picture into right SNS.

Item type markers

Item markers are created for indicating the type of the physical object. Currently the SPS application is supporting the following types of markers: Toy, Jewelry, Gadget and Electronic.

Like social network marker, the Item type markers are also restricted to having only one visible marker.

Empty markers

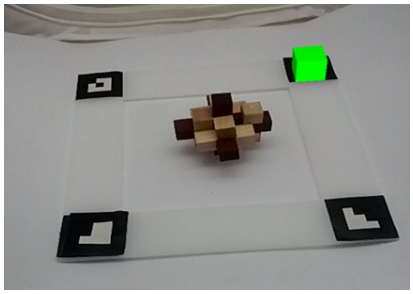
The main purpose of this marker type is to allow calculations of physical object size in the future. In current state this type of marker is used just for hiding the marker using blank texture. Naturally this texture can be changed if needed into any type of texture.

GRAPH 6. Example of SNS and Item type of markers

4.4 AndAR

As stated earlier the chosen library for the Smart Picture Studio project, was the AndAR library. This chapter explains how the AndAR was used in the project and gives a short explanation on how the architecture is built.

It should be noted that the information which is presented in this chapter might differ from the current version. The AndAR project is active and even the architecture of the project may change. The latest version can be found on the project website. (http://code.google.com/p/andar/)

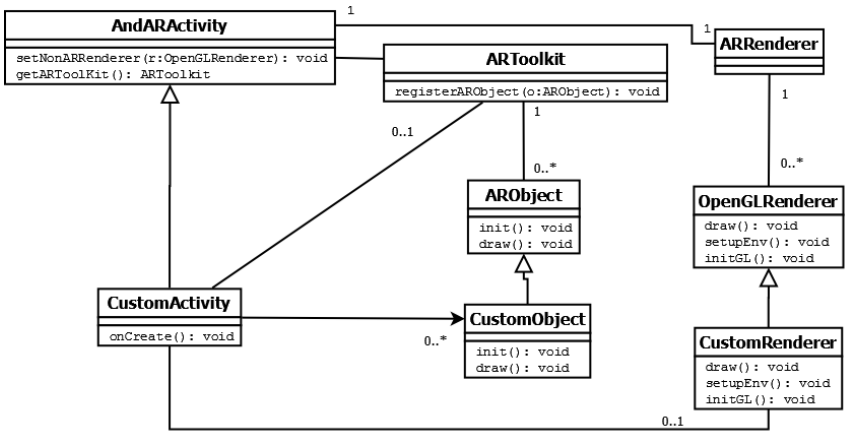


GRAPH 7. AndAR's example application in use

4.4.1 Architecture

Main usages of the AndAR library can be divided on three classes. These classes should not be used directly, because they are created for inheritance purposes only. The following graph is taken from the documentation and gives fair view of the architecture of AndAR.

It should be noted, that in the example that is given in form of a graph, the classes are following the naming policy of having the word "Custom" in the beginning of the name. For example chapter "4.4.2 Creation of inherited classes" presents class named SPSARactivity, in the following UML diagram; this class would take place of CustomActivity. (AndAR 2012.)



GRAPH 8. UML Diagram of AndAR (adapted from AndAR, 2012)

AndARActivity

When applications are developed for the Android platform, the normal way is to create one or more activities. For developing application which uses AndAR framework, it is necessary to create a class that is inherited from this abstract class. The class itself does all necessary work behind the curtains for example handling the camera, marker detection and displaying the video stream.

ARRenderer and OpenGLRenderer

The ARRenderer class is responsible for everything, that is related to OpenGL. It can be used for implementing OpenGLRenderer. This allows mixing non augmented with augmented 3D objects. The main purpose of this class is to allow developer to access on certain settings for example lighting and another general purpose OpenGL related initializations.

ARObject

ARObject class is responsible for holding the information about the marker. Because the framework is based on ARToolKit, it can use the same data patterns which are generated for the basic ARToolKit. The framework has one limitation. This limitation is based on how the framework is loading the data which is used for recognition of markers. For storing the data, this data needs to be saved as files into Android projects asset folder. This is necessary for finding the files which are presented when creating the object.

4.4.2 Creation of inherited classes

As stated in the previous chapter, the AndAR is used by creating classes that are inheriting the parent classes which are presented in earlier UML diagram.

Inside this chapter these classes that were used for creating the Smart Picture Studio are presented shortly with small description of the important parts for understanding the source code.

SPSARActivity

The SPSARActivity class can be described as most important class for understanding the basic usage of the AndAR. The usage of this activity class is not complex, mainly because the parent class is holding all the complexity inside. This makes the usage of the class quite simple and easy.

```
artoolkit = super.getArtoolkit();
```

At first, the reference of ARToolKit's object is taken by using the super class method getArtoolkit. After this the reference for the object is taken, it can be used for registering new ARObjects.

```
artoolkit.registerARObject(
           new SNSARObject(SNSTypes.Flickr,"000.patt", this)
           );
```

The code snipped above is simply registering the ARObject type of object. For saving the space, the new SNSARObject is created when it is given as parameter. The reason for this is the large amount of different types which are needed to support. When the SNSARObject is created, the constructor method is taking few parameters: Type of the social network, the pattern file and reference for the activity. It should be noted, that reference to the activity is used for accessing texture data.

SNSARObject and ItemARObject

Both of these classes are inheriting the ARObject class. Their purpose is to inform a user about AR marker that is set in the studio box. Both of the classes are also

following the same pattern. In code snipp below, the part of rendering and setting the item type is presented as example. The code itself is from ItemARObject class.

```
if(!mTypeMarked)
{
  mTypeMarked = true;
  MTypeManager.getInstance().setItemType(mItemType);
}
square.draw(gl);
```

When the marker is spotted first time, the type for manager is set. This occur only once in lifetime of the object. Below the structure the square object's draw method is called, for rendering the wanted type.

It should be noted that the enum variable named mItemType is set when object is registered at creation.

NormalARObject

The main duty of the NormalARObject is to have a possibility to do extra calculations on physical objects. The current system does not use these calculations and marker type is created for future use only.

The main difference between the earlier presented ARObject based classes is the loadTexture method. The code snipped that is presented below. It can be seen that the loadTexture method always returns the same resource as bitmap.

```
private Bitmap loadTexture(){
    Bitmap bitmap;
    bitmap = BitmapFactory.decodeResource
                (
                mContext.getResources(), R.drawable.mnormal
                );
   return bitmap;
 }
```

MTypeManager

MTypeManager is singleton type of class. The purpose of this class is to have a way of informing other parts of the application about selected social network and the physical objects type.

The MTypeManager is not really part of the augmentation, but the values for it are set by the ARObject classes. The following code snip is presents how the object of this class is called.

```
MtypeManager.getInstance().setSNSType(mSNSType);
```

It should be noted, that the usage of MTypeManager is not the official way of informing the newly found marker. The reason why this class were created, is the missing part of AndAR's documentation, which did not explain how to use the callback functionality.

4.4.3 OpenGL

The rendering of 3D objects is handled by using Open Graphics Library (OpenGL). More accurately the ES version what is optimized for mobile devices. OpenGL is an open source project which has long development history, it was introduced at year 1993 and it has been under constant development after release. (OpenGL 2012.) Currently it is one of most widely used library for presenting 3D objects and environments. The main reason for the wide usage in the industry is the reliability and portability for different platforms.

While the usage of OpenGL is important for using AndAR, the basic usage of the OpenGL is left outside from this thesis. The reason for this is the large amount of information which would need to be presented.

SPSRenderer

The OpenGLRenderer class presented earlier is used as base for the SPSRenderer class. In this SPS project it has been used only for setting the lighting settings. The following code snippet is presenting a setupEnv method that is used to control lighting settings.

```
public final void setupEnv(GL10 gl) {
            gl.glEnable(GL10.GL_LIGHTING);
            gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_AMBIENT,
            ambientLightBuffer1);
            gl.glLightfv(GL10.GL_LIGHT1, GL10.GL_DIFFUSE,
            diffuseLightBuffer1);
            gl.glLightfv(GL10.GL_LIGHT1,
            GL10.GL_SPECULAR,specularLightBuffer1);
```

```
            gl.glLightfv(GL10.GL_LIGHT1,
            GL10.GL_POSITION,lightPositionBuffer1);
            gl.glEnable(GL10.GL_LIGHT1);
            initGL(gl);
           }
```

SPSSquare

SPSSquare is a simple class, which is creating a plane, where the image of marker type is set as a texture. The most visible duties of this class are loading the texture by using loadGLTexture method and rendering the model by using the draw method.

The code snipp below loads the bitmap which is given as parameter for the method, as bitmap object. By using OpenGL's functions the texture is generated by using the bitmap and it is bind for drawing, when draw method is called.

```
public void loadGLTexture(GL10 gl, Bitmap bitmap) {
 gl.glGenTextures(1, textures, 0);
 gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[0]);
 gl.glTexParameterf(GL10.GL_TEXTURE_2D,

 GL10.GL_TEXTURE_MIN_FILTER, GL10.GL_NEAREST);
 gl.glTexParameterf(GL10.GL_TEXTURE_2D,
  GL10.GL_TEXTURE_MAG_FILTER,
  GL10.GL_LINEAR);
 GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bitmap, 0);
 bitmap.recycle();
}
```

As stated before, the draw method is responsible for actual rendering of the object. Before this method can be called, the texture needs to be loaded by the method that was presented as code snipped previously.

```
public void draw(GL10 gl) {
 gl.glBindTexture(GL10.GL_TEXTURE_2D, textures[0]);
 gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
 gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
 gl.glFrontFace(GL10.GL_CW);
 gl.glVertexPointer(3, GL10.GL_FLOAT, 0, vertexBuffer);
 gl.glTexCoordPointer(2, GL10.GL_FLOAT, 0, textureBuffer);
 gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, vertices.length / 3);
            gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
            gl.glDisableClientState(GL10.GL_TEXTURE_COORD_ARRAY);
}
```

The draw method itself is quite simple. It loads the earlier given textures and presents them, when the method is called.

5 CONCLUSIONS

The actual development of Smart Picture Studio did not have major problems. Still there were two minor problems what were caused by the chosen library called AndAR. These problems were fixed by modifying the actual code of library. It should be noted, that these problems were caused by newest development version and they might not appear in latest version.

When researching the Augmented Reality it was found that there is a lot potential existing. The biggest challenge in the future is probably hardware. While algorithms are evolving towards more complexity and can create augmentation without the need of AR markers, the hardware is creating problems as for size and implementation.

After researching Augmented Reality, the following conclusion was made. Even with all problems that are existing within current technology, idealistic situation or science fiction can be closer than general public thinks. Just a while ago, Google announced information about upcoming AR glasses, which allow the user to interact with the environment that he sees. The actual technical information about these glasses has not been released yet. Still, this is a good example of big company that is not going to be left behind in race of Augmented Reality.

REFERENCES

A9 2012. Available: http://a9.com/-/company/flow.jsp . Accessed 15 May 2012.

APT 2012. Available: http://packages.qa.debian.org/a/apt.html. Accessed 16 May 2012.

ARToolKit 2012a. Available: http://www.hitl.washington.edu/artoolkit/documentation/. Accessed 15 May 2012.

ARToolKit 2012b. Available: http://artoolkit.sourceforge.net/apidoc/. Accessed 16 May 2012.

ARToolKitPlus 2012. Available: http://handheldar.icg.tugraz.at/artoolkitplus.php. Accessed 16 May 2012.

AndAR 2012. Available: http://code.google.com/p/andar/wiki/HowToBuildApplicationsBasedOnAndAR. Accessed 16 May 2012.

Android 2012a. Available: http://developer.android.com/sdk/ndk/index.html. Accessed 16 May 2012.

Android 2012b. Available: http://developer.android.com/resources/tutorials/hello-world.html. Accessed 16 May 2012.

Android 2012c. Available: http://developer.android.com/guide/basics/what-is-android.html. Accessed 16 May 2012.

ArUco 2012. Available: http://www.uco.es/investiga/grupos/ava/node/26. Accessed 16 May 2012.

Azuma, R. 1997, A Survey of Augmented Reality. Available: http://www.cs.unc.edu/~azuma/ARpresence.pdf. Accessed 15 May 2012.

CAE 2012. Available: http://www.cae.com/en/healthcare/promis.simulator.asp. Accessed 15 May 2012.

Eclipse 2012. Available: http://www.eclipse.org/. Accessed: 16 May 2012.

FLARToolKit 2012. Available: http://www.libspark.org/wiki/saqoosha/FLARToolKit/en. Accessed 16 May 2012.

Feiner, S., MacIntyre, B. & Hollerer, T. 1997. A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=629922. Accessed 15 May 2012.

Flash Tarotaro. 2012. Available: http://flash.tarotaro.org/blog/2008/12/14/artoolkit-marker-generator-online-released/. Accessed 16 May 2012.

GSTreamer 2012, Available: http://gstreamer.freedesktop.org/documentation/. Accessed: 16 May 2012.

Henderson, S. & Feiner, S. 2009. Evaluating the Benefits of Augmented Reality for Task Localization in Maintenance of an Armored Personnel Carrier Turret. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5336486. Accessed 15 May 2012.

Hoshi, T., Takashi, M., Nakatsuma, K. & Shinoda, H. 2009, Touchable Holography. Available: http://www.alab.t.u-tokyo.ac.jp/~star/pdf/2009SIGGRAPH.pdf. Accessed 15 May 2012.

Kato, H. & Billinghurst, M. 1999. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. Available: http://www.hitl.washington.edu/artoolkit/Papers/IWAR99.kato.pdf. Accessed 15 May 2012.

Kilgrad, M. 1996. The OpenGL Utility Toolkit (GLUT) Programming Interface (API Version 3). Available: http://www.opengl.org/documentation/specs/glut/glut-3.spec.pdf. Accessed 16 May 2012.

MyGoldenI 2012. Available: http://www.mygoldeni.com/. Accessed 15 May 2012.

NAVIG 2012. Available: http://navig.irit.fr/index.php?option=com_content&task=view&id=14&Itemid=31. Accessed 15 May 2012.

NBAndroid 2012. Available: http://www.nbandroid.org/. Accessed 16 May 2012.

Narumi, T., Kajinami, T., Tanikawa, T. & Hirose, M. 2010. Meta cookie. Available: http://dl.acm.org/citation.cfm?doid=1836821.1836839. Accessed 16 May 2012.

NetBeans 2012. Available: http://netbeans.org/. Accessed: 16 May 2012.

NotCot 2012. Available: http://www.notcot.com/archives/2009/01/legos-digital-b.php. Accessed 15 May 2012.

NyARToolKit 2012. Available: http://nyatla.jp/nyartoolkit/wp/. Accessed 16 May 2012.

OpenGL 2012a. Available: http://www.opengl.org/. Accessed 16 May 2012.

OpenGL 2012b. Available: http://www.opengl.org/sdk/docs/. Accessed 16 May 2012.

osgART 2012. http://www.osgart.org/wiki/index.php/Main_Page. Accessed 16 May 2012.

Pink tentacle 2012. Available: http://pinktentacle.com/2007/06/ntts-tangible-3d-display/. Accessed 15 May 2012.

Qualcomm 2012. Available: http://www.qualcomm.com/solutions/augmented-reality. Accessed 16 May 2012.

Rekimoto, J. & Nagao, N. 1995. The World Through the Computer: Computer Augmented Interaction with Real World Environments. Available: http://dl.acm.org/citation.cfm?id=215639&bnc=1. Accessed 15 May 2012.

SLARToolKit 2012. Available: http://slartoolkit.codeplex.com/. Accessed 16 May 2012.

Samsung 2012. Available: http://www.samsung.com/global/microsite/galaxys/html/specification.html. Accessed 16 May 2012.

Spectrum IEEE 2012. Available: http://spectrum.ieee.org/biomedical/bionics/augmented-reality-in-a-contact-lens/0. Accessed 15 May 2012.

Starbucks 2012 . Available: http://www.starbucks.com/coffeehouse/mobile-apps/starbucks-cup-magic. Accessed 15 May 2012.

Thomas, B., Close, B., Donoghue, J., Squires, J., Bondi, P., Morris, M. & Piekarski, W. 2000. ARQuake: An Outdoor/Indoor Augmented Reality First Person Application. Available: http://www.tinmith.net/papers/thomas-iswc-2000.pdf. Accessed 15 May 2012.

Vuzix 2012. Available: http://www.vuzix.com/ar/product_browser.html. Accessed 16 May 2012.

Wagner, D. & Schmalstieg D. 2003. First Steps Towards Handheld Augmented Reality. Available: http://www.ims.tuwien.ac.at/media/documents/publications/HandheldAR_ISWC03final.pdf. Accessed 15 May 2012.