Bachelor's thesis

Information Technology

Embended software

2015

Raine Kuusisto

CREATING AUGMENTED REALITY

TURUN AMMATTIKORKEAKOULU

TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embended software

March, 2015| 62

Instructor: Tiina Ferm

Raine Kuusisto

CREATING AUGMENTED REALITY

Augmented reality (AR) is used in different occasions in our daily lives. The idea of an electronic display and spectacles that overlays data to people came in 1901. Augmented reality revolution started in 1990, it is a combination of real life and virtual reality. The beginning of AR is being examined from the very beginning of it. In 1999, it gained momentum when Hirokazu Kato published the ARToolkit to the open source community. After this, augmented reality has evolved everywhere, including smart phone applications.

ARToolKit is being used to create augmented reality applications. It is a software library to create augmented reality scenes. The program uses marker detection. When the camera identifies markers, 3D objects will appear on the top of each marker. ARToolKit uses black squared markers when detecting these. The use of these markers gives more accurate marker detection results. The camera has been also calibrated by using ARToolKit programs to produce better tracking results.

The ARToolkit allows video capture tracking of the real world to be combined with the interaction of virtual 3D objects. The equipment used was a Levovo Z500 laptop and OKER 335 HD Web Cam. Microsoft Visual Studio 2010 was being used as a compiler when building ARToolKit solutions. After the AR programs had been built, the testing was carried out in 3 phases.

When creating AR scenes, different 3D modelling software were being used. Demonstrations and the testing shows how the marker detection works. It also shows how to build ARToolKit code, and how to make working AR scenes with animated 3D characters. The final testing shows how the camera’s and markers interaction works. The results were animated 3D characters on the top each created marker.

KEYWORDS:

AR, ARToolKit, Marker, VRML

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ
TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautetut ohjelmistot

Maaliskuu 2015 | 62

Ohjaaja: Tiina Ferm

Raine Kuusisto

CREATING AUGMENTED REALITY

Nykyään lisättyä todellisuutta käytetään monessa eri yhteydessä. Idea elektronisesta näytöstä ja virtuaalilaseista, jotka toisivat datan ihmisten tietoisuuteen, tuli vuonna 1901. AR:n (engl. augmented reality) vallankumous alkoi 1990. Se on yhdistelmä todellista maailmaa ja virtuaalitodellisuutta. Laajennettua todellisuutta tarkastellaan sen alkuajasta lähtien. Vuonna 1999 AR sai uutta näkyvyytä, kun Hirokazu Kato julkaisi ARToolKitin avoimen lähdekoodin kaikkien saataville. Tämän jälkeen AR on levinnyt kaikkialle, mukaan lukien älypuhelimien sovelluksiin.

ARToolKit on ohjelmistokirjasto, jota käytetään lisätyn todellisuuden tekemiseen ja kohtauksien luomiseen AR:n maailmaan. Ohjelma käyttää merkkipohjaista tunnistusta. Kameran tunnistaessaan merkkejä, 3D-objekteja ilmestyy niiden päälle. ARToolKit käyttää tunnistamiseen mustia neliskulmaisia merkkejä. Näitä merkkejä käytettäessä saadaan tarkempia merkkitunnistustuloksia. ARToolkitin ohjelmakoodia käytetään myös kameran kalibrointiin, jotta saataisiin tarkempia merkkien seurantatuloksia.

ARToolKit välittää reaaliaikaista videokuvaa seuraamalla ympäristöä liittäen siihen virtuaalisia 3D-objekteja. Laitteet, joita käytetään testaamiseen ovat kannettava Lenovo Z500 tietokone ja OKER 335 HD-nettikamera. Microsoftin Visual Studio 2010:tä käytetään ARToolKitin ohjelmakoodin rakentamiseen. Koodin rakentamisen jälkeen, se testattiin 3:ssa eri vaiheessa.

AR-kohtauksien rakentamiseen käytettiin eri 3D-mallinnusohjelmia. Tarkoituksena on näyttää kuinka ARToolKitin merkkien tunnistus toteutetaan. Samalla ohjeistetaan kuinka ARToolKit-ohjelmakoodi rakennetaan ja miten saadaan tuotettua toimivia AR-kohtauksia, jotka sisältävät animoituja 3D-hahmoja. Lopputestauksessa voidaan nähdä, että kameran ja merkkien välinen yhteys toimii. ARToolKittiä testattaessa, tuloksena saatiin animoituja 3D-virtuaalihahmoja, jotka liikkuivat merkkien päällä.

ASIASANAT:

AR, ARToolKit, Marker, VRML

CONTENT

PICTURES

TABLES

REFERENCES

LIST OF ABBREVIATIONS

API	Application Programming Interface
AR	Augmented Reality
ARToolKit	Software library to create Augmented Reality. It was originally developed by Dr. Hirokazu Kato
BVH	Biovision Hierarchy, charater animation file format
COM	Component Based Modelling
DAT	Computer filename extension, typically for a file considered to contain data
DMS	Dynamic Motion Synthesis
DSVL	Direct Show Video Library
FBX	Filmbox file format
GPU	Graphics Processing Unit
GLUT	Graphics Library Utility Toolkit
GUI	Graphic User Interface
HIT Lab	Human Interface Technology Laboratory
HMD	Head Mounted Display
KARMA	Knowledge-based Augmented Reality for Maintenance Assistance
OS	Operating System
SLN	Microsoft Visual Studio solution file
SSD	Sum of Squared Differences
VR	Virtual Reality
VRML	Virtual Reality Modelling Language
OpenGL	Open Graphics Library

1 INTRODUCTION

Augmented reality (AR) is used in many occasions in our daily lives. Augmented reality applications can be seen these days everywhere. Applications are used to combine real life to virtual environment. AR technology is commonly used in manufacturing processes. AR can be a live, direct or indirect, view of a physical, real-world environment. Elements that are mentioned earlier are augmented by computer-generated sensory input such as sound, video graphics or GPS data. This is how an augmented reality system generates a composite view for the user. The view is a combination of the real scene by the user and a virtual scene generated by the computer that augments the scene with additional information. [1] [2]

In the year 1999, ARToolKit allows video capture tracking of the real world to combine with the interaction of virtual objects and provided 3D graphics that could be overlaid on any OS platform. AR has many potential applications in the field of industrial and academic research. [2] In this Bachelor's thesis, Dr. Hirokazu Kato's ARToolKit software library is being used to build AR applications in Windows 7 environment.

ARToolKit is a C and C++ language software library. Programmers can easily develop augmented reality applications with these software library files. Software library contains ARToolKit's C language source code and header files. In this project, source and header files are built by using Microsoft Visual Studio 2010 as a compiler. Visual Studio 2010 coverts source code into another language. It is usually called as machine code or machine language so it can be directly understood by processors.

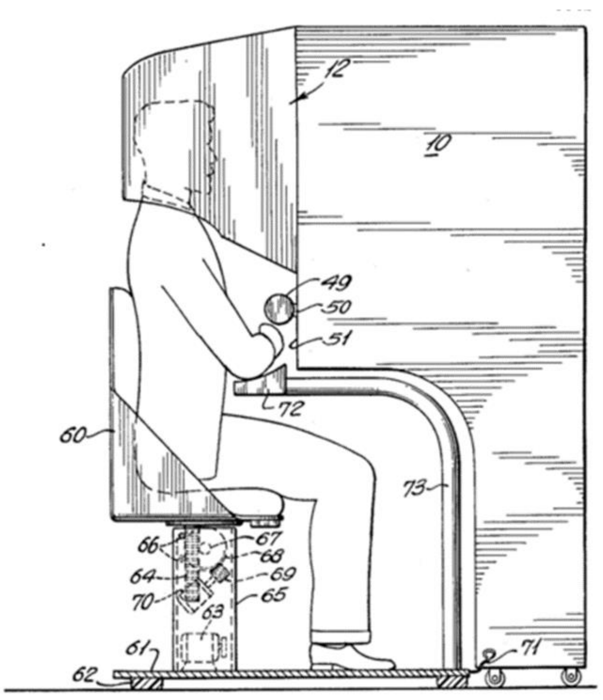
In this Bachelor's thesis, marker detection is used to create augmented reality. Different markers are being used to show markers and camera's interaction. Equipment that are used is Lenovo Z500 computer and OKER HD 335 HD Web Cam. When the camera sees different markers, 3D characters with animations are shown.

The character is created by using an open source program called Makehuman, the produced 3D model is exported as an object (OBJ) file. The object file is imported to Cinema 4D to create character's animations. A program called Endorphin is used to make joints and movements for the character. The 3D human character and joints are being combined using Cinema 4D. Autodesk 3ds 2015 is being used to finalize functional Virtual Reality Modelling Language (VRML) scene with animated character. Finally, the character with different movements are exported as VRML files and moved to ARToolKit's wrl directory.

In this project, testing comes in 3 phases. At first, ARToolKit examples are being tested so it can been seen that the code is working. The 2nd testing take a place when animated character is created with Cinema 4D and exported as VRML files. The 3rd testing is done when VRML files are converted from Filmbox (FBX) format and exported to VRML scene by using Autodesk 3ds 2015. The final experimentations show scenes with animated 3D characters appearing on the top of each marker. The results are successfully detectected markers and working animated VRML scenes.

2 BEGINNING OF AUGMENTED REALITY

AR was invented since mankind started to make different gadgets that could relate to their environment and supply their users with information based on that. In 1901, L. Frank Baum was the first person who mentioned the idea of an electronic display and spectacles that overlays data to the people [3]. In the year 1957, a cinematographer Morton Helig began to build a machine called the Sensorama. It was designed as a cinematic experience to take in all your senses. The Sensorama looked like an arcade machine from the 80s (Picture 1). It blew wind at you, had vibrated seat, played sounds and projected a form of a stereoscopic 3D environment to the front and sides of your head. [4].



Picture 1. The Sensorama, from U.S. Patent #3050870

In 1966, Professor Ivan Sutherland of Electrical Engineering at Harvard University invented the first model of the head mounted display (HMD) and in 1968 Sutherland invents a head-mounted three-dimensional display. These days, head mounted display is one of the most important devices used in both AR and virtual reality (VR). This was the first step to making AR usable possible

[5]. In 1975, Myron Krueger created artificial reality lab called the Videoplace. There people could interact with virtual objects for the first time.

It was year 1980 when Steven Mann creates the first wearable computer. It was a computer vision system with text and graphical overlays on a photographically mediated reality [6]. Dan Reitan brought geospatially maps, multiple weather radar images and space-based and studio cameras to virtual reality Earth maps and abstract symbols for television weather broadcasts in 1981. This brought augmented reality to be broadcasted on TV [7]. An American writer and a multi talent musician called Jaron Lanier brought out the phrase virtual reality in 1989. Jaron Lanier creates the first commercial business around virtual worlds [8].

3 AR GETS MEANING

In the year 1990, the words augmented reality gets meaning by Professor Tom Caudell while he was working in Boeing’s Computer Services' Adaptive Neural Systems Research and Development project in Seattle. In a search to find an easier way to help the aviation company’s manufacturing and engineering process, he began to apply virtual reality technology to maintenance. This led to invention of a complex software that could overlay the positions of where certain cables in the building process were supposed to go. [9] [10]

In 1991-1992, two other teams made breaktrought in the field of AR. Louis Rosenberg creates what is widely recognized as the first functioning AR system for the US Air Force known as Virtual Fixtures. He also published the first study of how an AR system can enhance human performance. [10]

The second group of researchers consisted of Steven Feiner, Blair MacIntyre and Doree Seligmann who are all leaders in the field of AR. They brought out an idea of a prototype system that they called KARMA (Knowledge-based Augmented Reality for Maintenance Assistance). The team from Columbia University built an HMD with Logitech-made trackers attached to it. Their project’s idea was to develop 3D graphics of a ghost image to show people how to load and service the machine without having to refer to instructions. [10]

AR came worldly known in 1994 when Julie Martin became the first person who brought the concept into public performance. She created a government-funded show in Australia. It was like an ITV celeb show called Dancing in Cyberspace. The show consisted of dancers and acrobats who interacted with virtual objects projected into the same physical space as themselves. [10] In the year 1998, spatial augmented reality (SAR) was introduced at University of North Carolina at Chapel Hill by Ramesh Raskar, Welch, Henry Fuchs. In SAR, physical objects are augmented with images that are integrated directly in the user’s environment, not simply in their visual field. This way images could be projected onto real objects using light projectors, or embedded directly in the environment with flat panel displays [11].

Until 1999, AR was only used by scientists because it was expensive to carry out. Equipment were very large and softwares were complicated [10]. That is why AR did not reach consumers. Everything changed, when Hirokazu Kato of the Nara Institute of Science and Technology released the ARToolKit to the open source community. It allowed video capture tracking of the real world to combine with the interaction of virtual objects and provided a 3D graphics that could be overlaid on any OS platform. [2]

Smart phones were not invented yet. In this time, there was only a simple handheld gadget. It was equipped with camera and Internet connection. This brought AR to the people’s knowledge. [10]

In 2000, there was another AR revolution in the consumer area. Bruce Thomas and his team in the Wearable Computer Lab at the University of South Australia demonstrated the first outdoor mobile augmented reality video game. The game’s idea was to remove all the monsters and guns and place them on the top of real environment were users could move around without using any kind of joystick. ARQuake was born. A computer backpack, gyroscopes and GPS sensors were needed for gaming. When HMD was flipped down, users could find themselves in the parking lot of the University of South Australia teaming with demons on their way to kill each other. [10]

Years later in 2008, the first AR application came to smartphones. It brought Wikitude to the T-Mobile G1. This allowed Android users to take in the world through their mobile phone cameras and see augmentations on the screen of points of interest nearby. Wikitude came available to iPhone and Symbian platforms and later they launched an AR navigation application called Wikitude Drive. ARToolkit was also ported to Adobe Flash. This brought developers and consumers closer to AR world through a desktop browser and webcam. [10]

In the year 2013, Google announces an open beta test of Google Glass augmented reality glasses which became available to the public on May 15, 2014. The glasses reach the Internet through Bluetooth, which connects to the wireless service on a user’s cellphone. The glasses respond when a user speaks, touches the frame or moves the head [12]. In 2015, Microsoft introduces Windows Holographic and HoloLens augmented reality headset [13].

4 THE EQUIPMENT SPECIFICATIONS

Lenovo Z500 computer specifications (Table 1):

Table 1. Lenovo Z500 specifications

Model Name:	Lenovo IdeaPad Z500/P500
Machine Type:	20202, 5931 / 20210, 6279
Form Factor:	Dimensions Appr. 375.0 mm × 263.0 mm × 25.4 mm Weight Appr. 2.2 kg with 4-cell battery LCD size 15.6-inch
Processor:	Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz 2.10 GHz
Memory:	DDR3 2 x 4GB Slots SO-DIMM × 2
Hard disk drive:	Form factor 2.5-inch, 9.5 mm or 2.5-inch, 7.0 mm Interface SATA
Optical drive:	Form factor 9.5 mm, type Rambo
Display:	Display resolution (LCD) 1,366 × 768 pixels LCD backlight LED
I/O Ports:	USB 2.0 × 2, USB 3.0 × 1 Audio Combo audio jack × 1 Communication RJ-45 × 1 Video/Audio HDMI × 1 Video VGA × 1 Card reader 2 in 1 slot × 1 (SD/MMC)
Battery pack:	Type Li-ion cylinder battery (Cells/Capacity 4 cell, 41.6 Wh or 48 Wh)
AC adapter:	Input 100 - 240 V, 50 - 60 Hz AC Output voltage 20 V DC Power 65 W or 90 W
Miscellaneous:	Camera 720P HD or 0.3 Mega Security Kensington slot × 1

OKER HD 335 HD Web Camera specifications (Table 2):

Table 2. OKER HD 335 HD Web Cam

Model Name:	OKER HD 355 HD Web Cam
Video resolution:	1280 × 720 pixels
Display resolutions:	up to 12 M pixels
Frame rate:	30 fps
Support:	USB 1.1/2.0
Focus range:	30 mm-infinitives
Noise rate:	48db

5 ARTOOLKIT

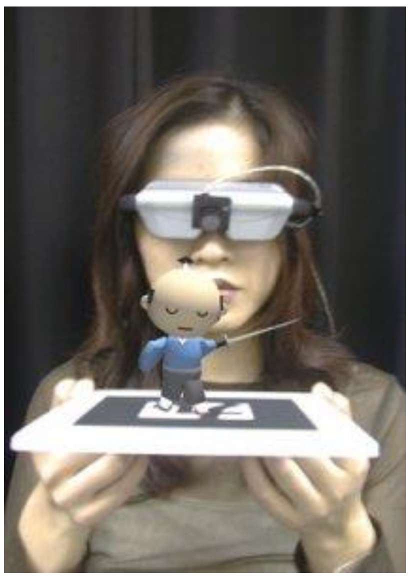
5.1 Introduction

The C programming language is the most frequently used language for writing operating systems, other programming languages and compilers. It is also often used for writing application programs. ARToolKit is also written in C/C++ language. ARToolKit was originally developed by Dr. Hirokazu Kato. After this, development is being supported by the Human Interface Technology Laboratory (HIT Lab) at the University of Washington, HIT Lab NZ at the University of Canterbury, New Zealand, and ARToolworks, Inc, Seattle [14].

ARToolKit is a software library for building AR applications. These applications involves the overlay of virtual imagery on the real world. Different markers are used to display virtual character and its animations.

OKER HD 335 HD Web Cam is being used to detect different markers (Picture 2). A different way to see a three-dimensional character is to use head set

Picture 2. OKER 335 HD Web Cam


Picture 3. Head set display

display. When a user moves a marker, the virtual character moves with it and appears attached to the real object (Picture 3).

One of the key difficulties in developing augmented reality applications is the problem of tracking the user’s viewpoint. In order to know from what viewpoint to draw the virtual imagery, the application needs to know where the user is looking in the real world. [14]

ARToolKit uses computer vision algorithms to solve this problem. The ARToolKit video tracking libraries calculate the real camera position and orientation relative to physical markers in real time. This software library gives an easy way to develop many different AR applications. Some of the features that are included in ARToolKit, these are directly retrieved from ARToolKit site [14]:

- Single camera position/orientation tracking.
- Tracking code that uses simple black squares.
- The ability to use any square marker patterns.
- Easy camera calibration code.
- Fast enough for real time AR applications.
- SGI IRIX, Linux, Mac OS and Windows OS distributions.
- Distributed with complete source code.

5.2 Installing ARToolKit

The ARToolKit Library is available for various operating systems like the Windows, Linux, SGI and MacOS X. In this Bachelor’s thesis, used operating system is Windows 7. ARToolKit can be installed with OpenGL graphics support, or with the OpenVRML renderer [14]. VRML support is needed because it allows to load and view VRML 97/2.0 models and animations.

ARToolkit can be in be installed into any directory. ARToolkit is installed and extracted into C:\ARToolkit folder. Used version of ARToolkit is 2.72.1-bin-win32.zip package. Before building ARToolkit in Windows 7 environment, certain prerequisites needs to be fulfilled (Table 3).

The computer and accessories must be able to handle video stream. A spare CPU is needed to calculate the tasks of video processing and display. To avoid compiler and linker errors, basic software dependencies are needed (Table 3). [15]

Table 3. ARToolKit prerequisites for Windows.

Prerequisite	**Instructions**
Development environment	Microsoft Visual Studio 6 and Microsoft Visual Studio .NET 2003 are supported, but it is also possible to build the toolkit using free development environments (e.g. Cygwin, http://www.cygwin.com/)
DSVideoLib-0.0.8b-win32	On Windows, DSVideoLib is used to handle communication with the camera driver. DSVideoLib-0.0.8b or later is required for ARToolKit 2.71. A source + binary package of DSVideoLib is included on the ARToolKit downloads page on sourceforge.
GLUT	Verify that GLUT runtime and SDK is installed. A binary package containing GLUT for Windows can be downloaded from http://www.xmission.com/~nate/glut.html. Verify that you have the GLUT runtime installed in your system directory. e.g. C:\Windows\System32 or C:\Windows\SysWOW64 when using 64 bit operating system • glut32.dll Verify that GLUT SDK is installed in your Visual C++ installation: • Include\gl\glut.h • Lib\glut32.lib
DirectX Runtime	Verify that DirectX runtime is installed: with Windows XP it is installed by default. You need to check your version; it must be 9.0b or later.
Video input device	Plug your camera or video input into your PC and install any necessary drivers. Verify that your camera has a VFW or

	WDM driver by running the program amcap.exe
(Optional, for VRML renderer only) OpenVRML-0.14.3-win32	A source + binary package of OpenVRML is included on the ARToolKit downloads page on sourceforge

These are installed on the computer:

- OpenGL Driver
- The GLUT library
- Microsoft's DirectShow (Direct X)

5.2.1 OpenGL Driver

Open Graphics Library (OpenGL) driver is a video card driver. Most of the graphic cards supports OpenGL. Drivers can be found from video card manufacturer’s driver page. OpenGL was originally developed by Silicon Graphics in the early '90s, OpenGL® has become the most widely-used open graphics standard in the world. It is a cross-language, multi-platform Application Programming Interface (API) for rendering 2D and 3D vector graphics. [16] In this thesis, NVIDIA GeForce GT 740 graphic card is being used. NVIDIA gives completely support to OpenGL and it is also being designed to give maximum performance to a computer’s Graphic Prosessing Unit (GPU).

5.2.2 The GLUT Library

GLUT is an OpenGL Utility Toolkit. It is a window system independent toolkit for writing OpenGL programs and an interface to graphics hardware. GLUT is a complete API written by Mark Kilgard. GL stands for Graphics Library and it is mainly a low-level graphics library specification. The library provides commands for specifying geometric objects in two or three dimensions. Toolkit also controls objects that are drawn on the display. In this case, objects are points, lines, polygons, images, and bitmaps. The information that GLUT supports are directly retvieved from the Internet site. [17]

The OpenGL Utility Toolkit supports:

- Multiple windows for OpenGL rendering
- Callback driven event processing
- Sophisticated input devices
- An 'idle' routine and timers
- A simple, cascading pop-up menu facility
- Utility routines to generate various solid and wire frame objects
- Support for bitmap and stroke fonts
- Miscellaneous window management functions

These library, header and dll files are used to compile the solution files correctly:

- glu32.dll
- glut32.dll
- opengl32.dll
- glu32.lib
- glut.lib
- glut32.lib
- GL.H
- GLU.H
- glut.h

The glut library, header and dll files are installed into these directories:

1. Opengl32.dll, glu32.dll and glut32.dll files have to be copied into Windows system directory. In this thesis, files are copied into C:\Windows\System32 and C:\Windows\SysWOW64.
2. GL.H, GLU.H and glut.h header files are copied into C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\GL.

3. Glu32.lib, glut32.lib and opengl32.lib library files are copied into C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\lib.

ARToolKit is extracted to C:\ARToolKit directory. Some modifications are made after extracting it. C:\ARToolKit\DSVL folder and files are being replaced with DSVL-0.0.8b.zip package files. After this, all the executable files that are located in the C:\ARToolkit\bin folder can be deleted because new versions of the executable files are built with Microsoft Visual Studio 2010. Microsoft runtime library files msvcp71d.dll and msvcr71d.dll are copied into C:\ARToolKit\bin folder. The files DSVL.dll and DSVLd.dll in the C:\ARToolKit\DSVL\bin are also copied into C:\ARToolKit\bin folder. [15] The glut library, header and dll files are installed into these directories:

1. Opengl32.dll, glu32.dll and glut32.dll files have to be copied into Windows system directory. The files are copied into C:\Windows\System32 and C:\Windows\SysWOW64.
2. GL.H, GLU.H and glut.h header files are copied into C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include\GL.
3. Glu32.lib, glut32.lib and opengl32.lib library files are copied into C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\lib.

5.2.3 Microsoft DirectShow (DirectX)

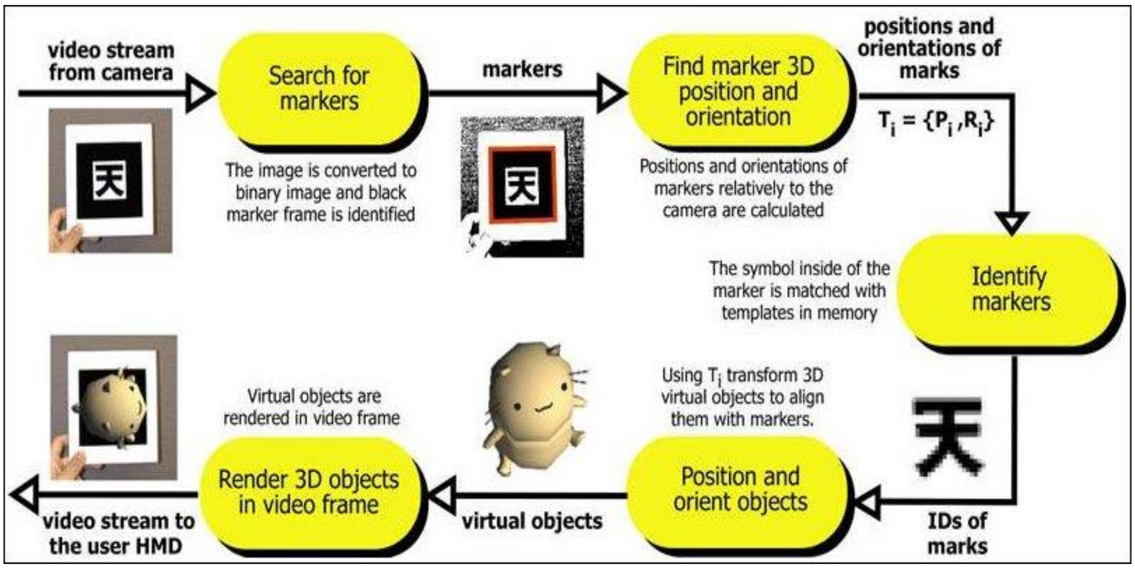
DirectX is an application program interface (API). The application can be used for creating and managing graphic images and multimedia effects in applications such as games or active web pages that will run in Microsoft's Windows operating system. It is based on the Component Based Modelling (COM). [18] [19] DirectX is a set of standard commands and functions that software developers can use when creating their programs. It can be used for controlling video playback, sound effects, and peripheral input such as a keyboard, mouse, or joystick [19].

6 ARTOOLKIT PRINCIPLES

ARToolKit applications allow virtual imagery to be superimposed over live video of the real world [20]. ARToolKit uses black squares when tracking markers. These markers are called as template markers. The ARToolKit tracking follows these principles which are directly retrieved from ARToolKit site [20]:

1. The camera captures video of the real world and sends it to the computer.
2. Software on the computer searches through each video frame for any square shapes.
3. If a square is found, the software uses some mathematics to calculate the position of the camera relative to the black square.
4. Once the position of the camera is known a computer graphics model is drawn from that same position.
5. This model is drawn on top of the video of the real world and so appears stuck on the square marker.
6. The final output is shown back in the handheld display, so when the user looks through the display they see graphics overlaid on the real world.

The Picture 4 summarizes steps that ARToolKit uses. ARToolKit is able to perform camera tracking in real time, ensuring that the virtual objects always appear overlaid on the tracking markers [20].


video stream from camera
Search for markers
markers
Find marker 3D position and orientation
positions and orientations of marks
$T_i = \{P_i, R_i\}$
The image is converted to binary image and black marker frame is identified
Positions and orientations of markers relatively to the camera are calculated
The symbol inside of the marker is matched with templates in memory
Identify markers
Virtual objects are rendered in video frame
Using T_i transform 3D virtual objects to align them with markers.
Render 3D objects in video frame
virtual objects
Position and orient objects
video stream to the user HMD
IDs of marks

Picture 4. ARToolKit principles.

7 MARKER-BASED TRACKING

Marker-based tracking is being used in this Bachelor's thesis. This means that the system needs to know where the used camera is being pointed at. When pointing camera to the markers, system is checking the location and orientation of the camera. Camera calibration is used to get better results when the camera is pointed to the markers. Calibration also makes system to be able to show virtual character in the correct place. The word tracking basically means that the camera's location and orientation is calculated in real time [20]. This is one of the basic components when creating augmented reality.

Usually, most of the augmented reality systems already have integrated camera installed in as part of the system. Visual tracking methods are usually related to AR and this thesis is also approached by using marker-based tracking method.

The main concentration is in visual tracking, because type of this project and the system that is being used. When the camera is observing environment, the system is selecting orientation of the coordinates randomly. This makes user unable to get correct visual observations. Marker-based solution is being used to get a simple way to computer vision technique to find printed out markers. A marker can be a simple sign or an image. The computer vision detects these markers from a live video feed. When detecting markers, system uses image processing, pattern recognition and computer vision techniques. The computer vision detects the marker and scales it correctly and also defines correct pose of the camera. [22]

Simple markers are being used because these give better and reliable detection. Differences in brightness are more easily to detect than color differences when using machine vision techniques. Usually, different cameras do not display white balance correctly. The camera may not see colors correctly if there is something else in the picture or lighting is bending the colors. [20] [21] When using simple markers which luminance is very high, the more easily objects are detected. The best detection of the markers can be done by using black and white markers.

Four known points are sufficient to calculate the pose of the camera uniquely and simplest way to get these points is to use black square markers [21]. Because of the fast and reliable detection, many systems use black and white square markers. These markers are also used to get good tracking results.

7.1 Detecting Markers

The used camera needs to find the outlines of the markers and then to gather locations of marker's corners in the image. If markers are found, the system confirms their identity. Markers pose are calculated by the used system. The system is using the information from the detected marker location when detecting the pose of the markers. A basic marker detection procedure follows these steps which are directly retrieved from the Internet [22]:

1. Image acquisition
2. Preprocessing
3. Marker detection
4. Identification and decoding
5. Calculating pose of the markers

7.2 Image Acquisition

The image acquisition provides the image for the marker detection. When the system is detecting markers, it can see markers correctly or may reject used marker at the any processing stage [22].

7.3 Preprosessing

Before markers are detected, the system needs to get a greyscale image of the used marker. When an image is being processed, the first task that the system needs to do is to find edges of the used marker. There are 2 ways how the system is using detection [2] [22]:

1. Thresholds an old image and search markers from the binary image
2. Detect edges from a greyscale image

At first, the system counts that all objects are potential markers. When the system finds potential markers, location of the markers are undistorted for line fitting. After this, the system tests all the seen markers and checks if markers have 4 straight lines and 4 corners. If the system finds square markers, it optimizes the corner locations in sub-pixel accuracy and these locations are used in further calculations. [22] [23] [24]

Edge detection in greyscale images is really time consuming and that is why sub-sampling method is usually being used. It detects only edges on a predefined grid. Using this method, the marker detection is linking edge pixels into segments. Usually, systems are grouping segments into longer lines using edge sorting. When the system samples 4 corner points using a course grid, 4 straight lines are extended to full length to find accurate corner points of the markers. The gradient information of the original image is being used when extending the edges to full length. [23] [24] There are different methods that applications are using for line detection, line fitting and line sorting. ARToolKit uses edge sorting method. [22]

7.4 Marker Detection

When the system is detecting 2D locations of edges and corners, even small errors in detection can affect the calculated pose of the camera. There are various reasons which can cause these errors like fast movement of the camera, incorrect threshold value, motion blur, noise ratio, etc. [25, 26, 27]

Augmented reality applications are targeting for real-time processing and fast performance. That is why programs are using many implementations to get to this point as fast as possible. It is important that the system rapidly counts off objects that are not real markers. The system can also reject markers if the camera is too far away or markers are too tiny or when a lighting is creating reflections that are chancing grayscale values [20]. [22]

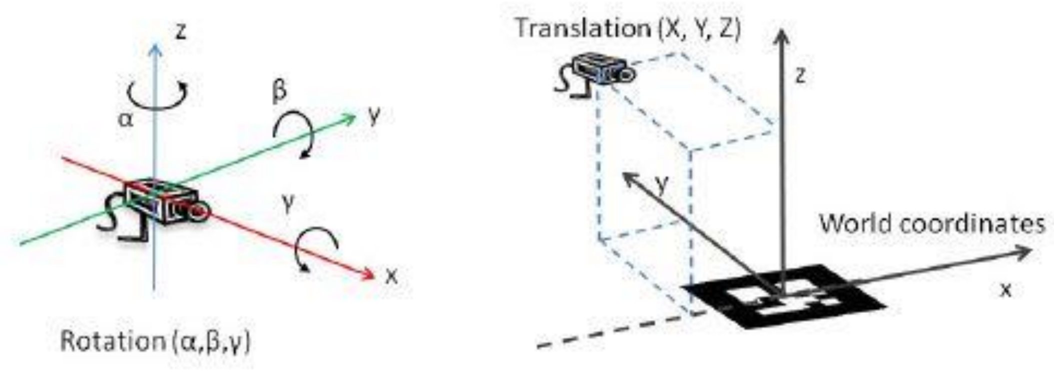
ARToolKit is one of those programs which assumes that the markers are detected from suitable distance of the camera. ARToolKit marker detection have areas that are too small or too big after labelling. This makes boundries when detecting markers. When tracking markers, there are also range issues. There are resuts shown in the ARToolKit site where can be seen how camera's distance and size of the markers affect when detecting these. Typical maximum ranges for square markers of different sizes (Table 4): [20]

Table 4. Tracking range for different sized markers.

Pattern Size (inches)	**Usable Range** (inches)
2,75	16
3,50	25
4,25	34
7,37	50

7.5 Calculating pose of the markers

When talking about a marker's pose, it refers to its location and orientation. The location of the markers can be shown with 3 translation coordinates (x, y, z) and orientation as 3 rotation angles (a, b, c) around the 3 coordinate axes (Picture 5). A pose also has 6 degrees of freedom (6 DOF). [22]

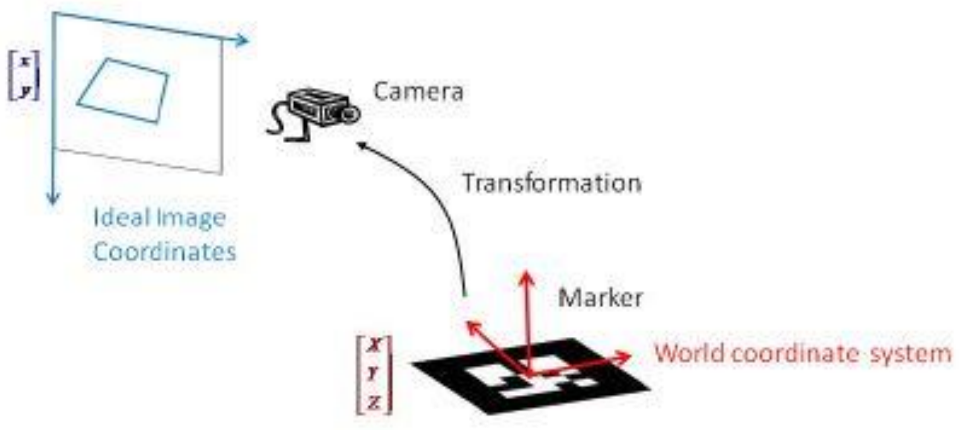


Picture 5. Camera's location and orientation in world coordinates.

The pose of the calibrated camera can be uniquely determined from a minimum of 4 coplanar but non-collinear points [21]. In this thesis, the system is calculating the markers poses in 3D coordinates using the 4 corner points of the markers [22]. The system goes through following procedures when calculating projective geometry and camera calibration.

An ideal pinhole camera model can be described when all rays pass the infinitely small optical centre of a camera and the object’s image registers on an image plane. In digital cameras, the image registers on the image sensor and coordinates of its elements differ from ideal coordinates. A camera’s image depends on the camera’s physical features. These features are camera’s focal length, image sensor orientation, size, etc. [2] [22]

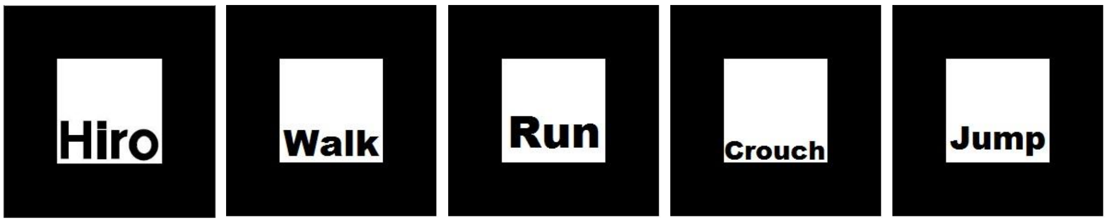
There are also equations which are used to calculate camera’s and marker’s relationships and other parameters [28]. When the camera is pointed to the markers, a tracking system is solving camera matrix for each marker when detecting these. A transform matrix converts world coordinates to ideal camera coordinates (Picture 6) [22].



Picture 6. Transform matrix conversion to ideal camera coordinates.

8 TEMPLATE MARKERS

Template markers are used in this Bachelor’s thesis. The first ARToolKit markers were template markers. Template markers are square black and white markers that have a simple picture or text inside of a black border (Picture 7).



Picture 7. The markers that are being used.

The system detects the markers by comparing their segmented images with marker templates. When the system is identifying markers, application matches markers that the camera sees and the best match defines its identity. The used markers have only text inside of the black squared markers.

When the marker detection process is identifying the markers, it defines the matching area. The location, size, and orientation of the matching area are unknown. Because template markers are being used, the detected markers are undistorted using the calculated camera pose and scaled to the same size as marker templates and compared in 4 different positions to all marker templates. The marker that gives the highest similarity value when detecting used markers, is the correct marker. If all the similarity value are lower than a threshold, the system rejects markers. [22]

The system can project the center areas of all cells on the used marker image into image coordinates using the calculated camera pose instead of unwarping the whole image. This way the pixel values can be sampled directly from the greyscale or threshold image. The value can be point to the nearest pixel, the average or mean of N nearest pixels or the value of all pixels inside the sampling grid cell, etc. Before the system is checking the template matching of a greyscale image, it is usually normalized. The system is normalizing images in a way that the darkest areas are black and the lightest are white. [2] [22]

For example, the similarity value of the markers can be based on SSD (the sum of squared differences) or cross correlation. SSD dissimilarity value can be calculated using this formula (Equation 1): [22]

Equation 1. The sum of squared differences.

$$D = \sum_i d(x_i, y_i)^2$$

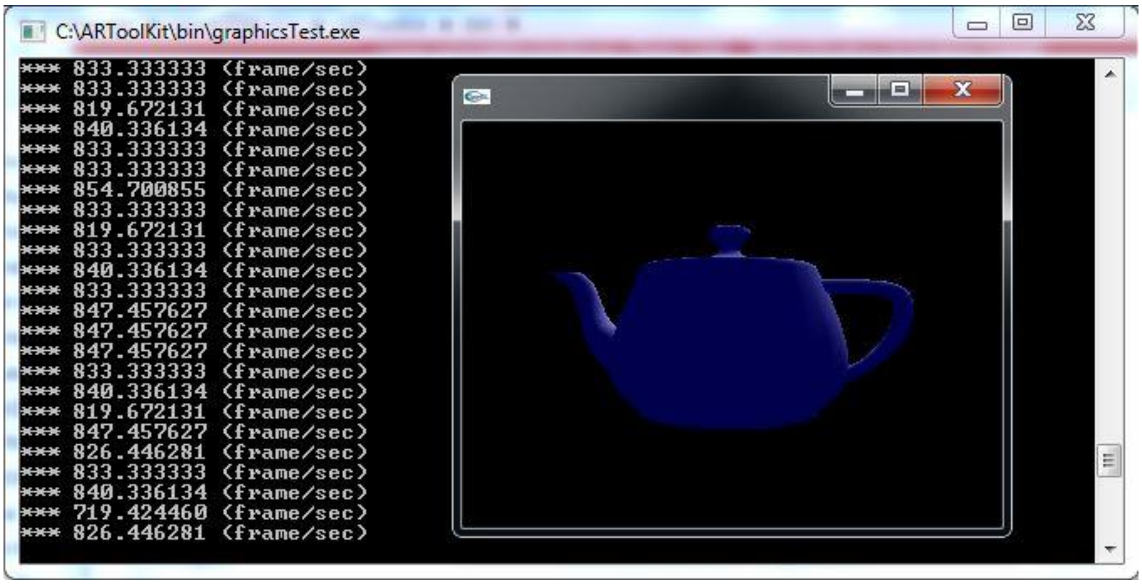
9 BUILDING ARTOOLKIT WITH VISUAL STUDIO 2010

In the C:\ARToolKit directory, Microsoft Visual Studio Solution (SLN) file ARToolkit.sln is being opened and built with Visual Studio 2010. The ARToolkit files should be built without errors, warnings can be ignored. Visual Studio output shows that 26 projects are built correctly. Two projects are skipped. These projects are built when installing ARToolKit's optional builds.

Markers are needed when starting the project. Different markers are being used to show markers and camera's interaction. A default Hiro marker is being used with sample applications and for testing purposes. The sample markers can be found in C:\ARToolkit\patterns folder. The C:\ARToolkit\patters folder also contains blank marker that can be used for designing own markers.

9.1 Testing Graphic and Video

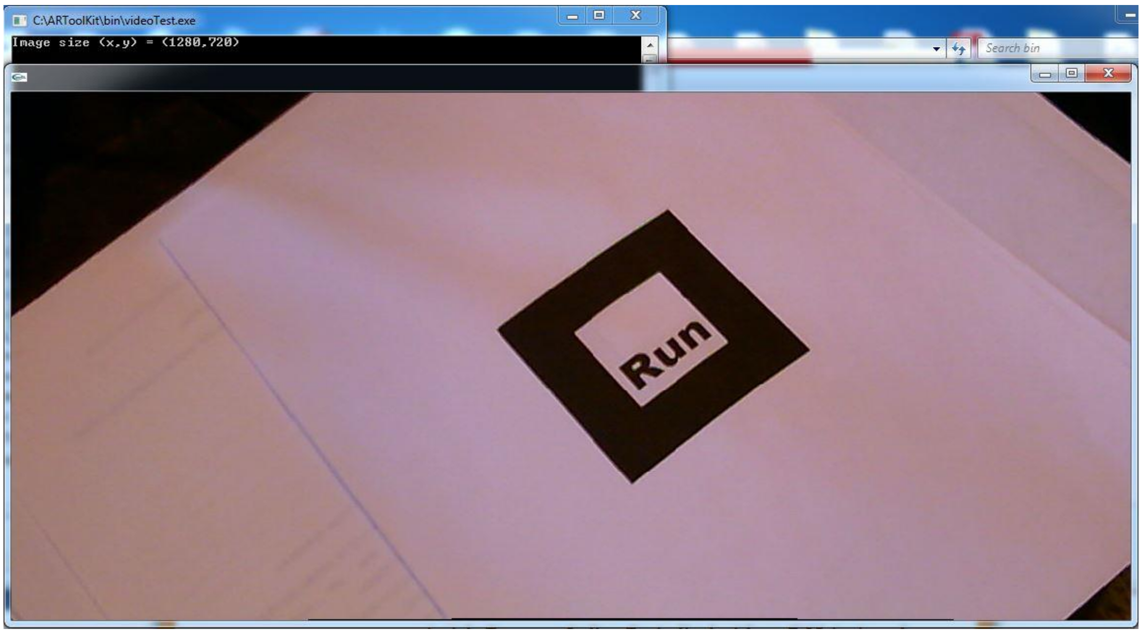
When the projects are built, the graphic and video can be tested. The graphic part is tested with graphicsTest.exe file which is located in C:\ARToolkit\bin folder [15]. The test shows that the graphical part is working. A window appears and shows a rotated 3D teapot (Picture 8).



```
*** 833.333333 (frame/sec)
*** 833.333333 (frame/sec)
*** 819.672131 (frame/sec)
*** 840.336134 (frame/sec)
*** 833.333333 (frame/sec)
*** 833.333333 (frame/sec)
*** 854.700855 (frame/sec)
*** 833.333333 (frame/sec)
*** 819.672131 (frame/sec)
*** 833.333333 (frame/sec)
*** 840.336134 (frame/sec)
*** 833.333333 (frame/sec)
*** 847.457627 (frame/sec)
*** 847.457627 (frame/sec)
*** 847.457627 (frame/sec)
*** 833.333333 (frame/sec)
*** 840.336134 (frame/sec)
*** 819.672131 (frame/sec)
*** 847.457627 (frame/sec)
*** 826.446281 (frame/sec)
*** 833.333333 (frame/sec)
*** 840.336134 (frame/sec)
*** 719.424460 (frame/sec)
*** 826.446281 (frame/sec)
```

Picture 8. Srceenshot of the rotating 3D teapot.

The second test is made to confirm that the OKER HD 335 HD Web Cam supports ARToolkit graphics module with OpenGL. A minimum frame rate of 15 fps (frames per second) is recommended, 30 fps is used and tested in this thesis. For testing and showing that video input works, videoTest.exe is executed from C:\ARToolkit\bin directory. The tests show that OKER HD 335 HD Web Cam supports ARToolkit video module and ARToolkit graphic module (Picture 9).



```
Image size (x,y) = (1280,720)
```

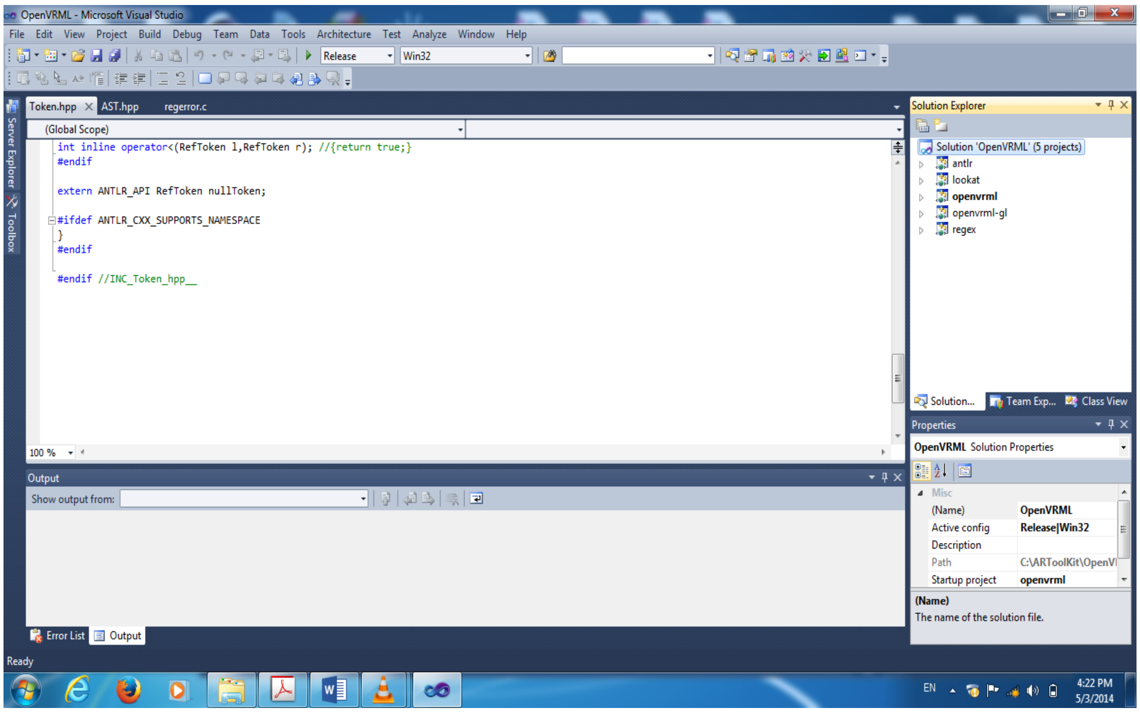
Picture 9. Screenshot of the video test.

9.2 Installing an Optional VRML Renderer

A zipped file openvrml-0.14.3.zip is being used. The zip package is unpacked into C:\ARToolKit\OpenVRML folder. OpenVRML.sln file is opened with Visual Studio 2010 from:

C:\ARToolKit\OpenVRML\src\openvrml-0.14.3\ide-projects\Windows\VisualC7\OpenVRML.

Visual Studio 2010 program starts and 5 projects are shown on Solution Explorer window (Picture 10).



```
int inline operator<(RefToken l,RefToken r); //{return true;}
#endif

extern ANTLR_API RefToken nullToken;

#ifdef ANTLR_CXX_SUPPORTS_NAMESPACE
}
#endif

#endif //INC_Token_hpp__
```

Properties: OpenVRML Solution Properties

Misc	
(Name)	OpenVRML
Active config	Release|Win32
Description	
Path	C:\ARToolKit\OpenVI
Startup project	openvrml

(Name)
The name of the solution file.

Picture 10. Screenshot of the opened SimpleVRML.sln file with Visual Studio 2010.

Before building the files, there are few changes that have to be made to these project files:

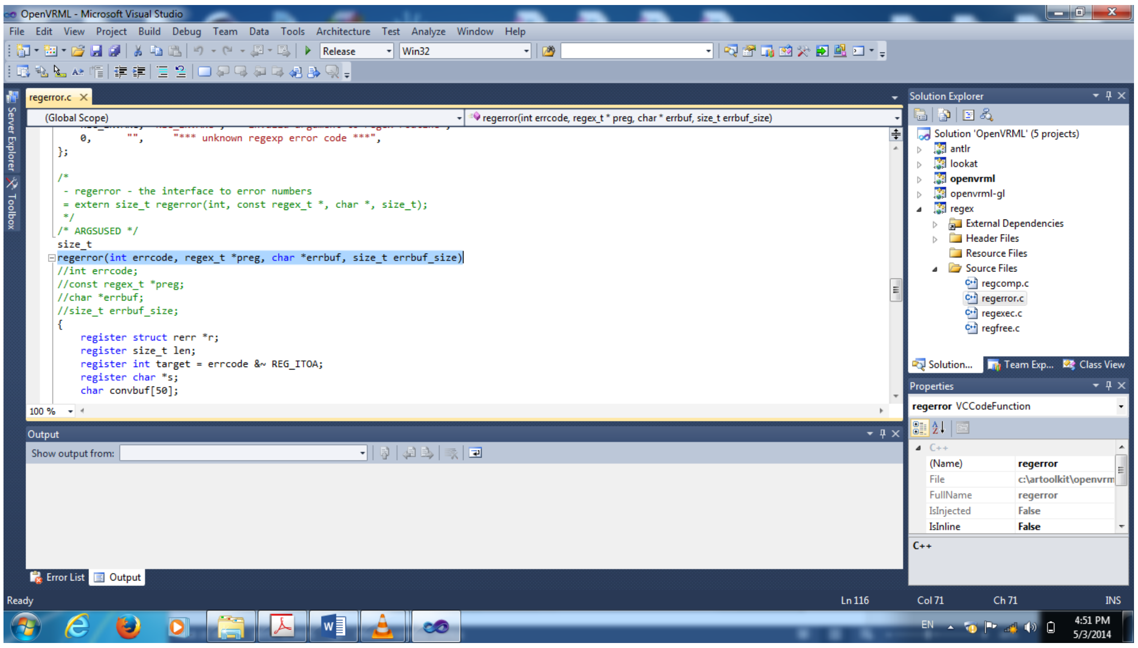
- regex
- antlr

The original regex project's regerror.c file:

```
regerror(errcode, preg, errbuf_size)
```

A modified regex project's regerror.c file (Picture 11):

```
regerror(int errcode, regex_t *preg, char *errbuf, size_t errbuf_size)
```

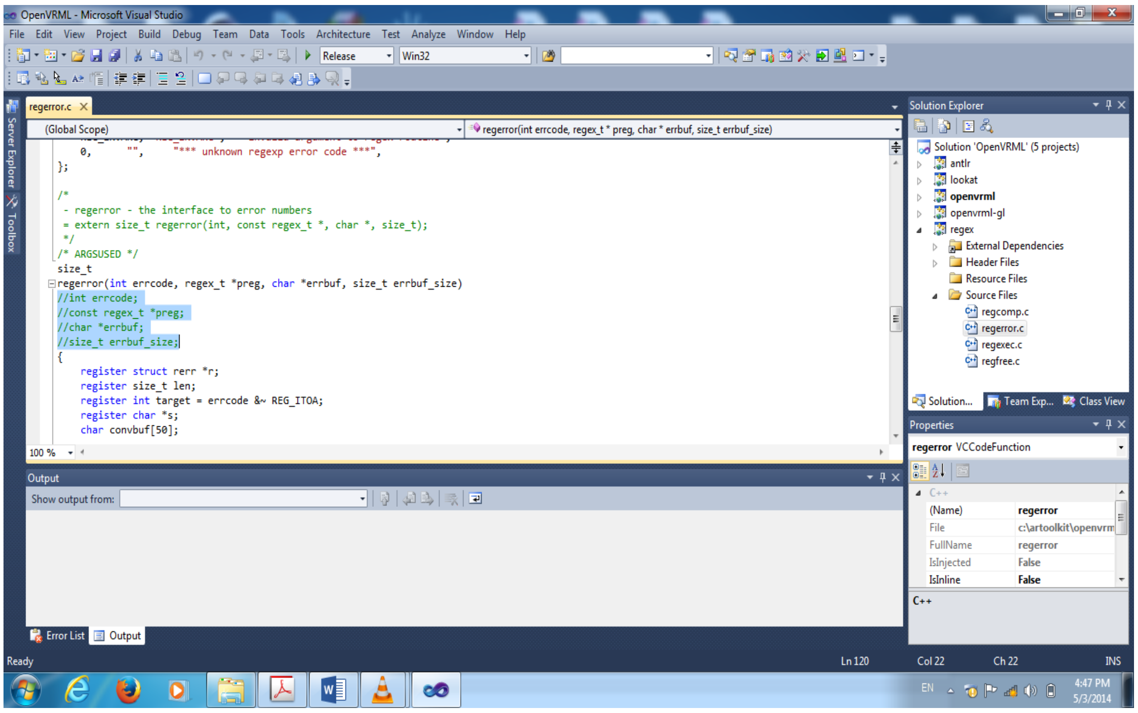


```
    0,      "",      "*** unknown regexp error code ***",
};

/*
 - regerror - the interface to error numbers
 = extern size_t regerror(int, const regex_t *, char *, size_t);
 */
/* ARGSUSED */
size_t
regerror(int errcode, regex_t *preg, char *errbuf, size_t errbuf_size)
//int errcode;
//const regex_t *preg;
//char *errbuf;
//size_t errbuf_size;
{
    register struct rerr *r;
    register size_t len;
    register int target = errcode &~ REG_ITOA;
    register char *s;
    char convbuf[50];
```

Picture 11. Editing the regerror.c code.

Four lines are commented (Picture 12):

```
//int errcode;
//const regex_t *preg;
//char *errbuf;
//size_t errbuf_size;
```



```
    0,       "",      "*** unknown regexp error code ***",
};

/*
 - regerror - the interface to error numbers
 = extern size_t regerror(int, const regex_t *, char *, size_t);
 */
/* ARGSUSED */
size_t
regerror(int errcode, regex_t *preg, char *errbuf, size_t errbuf_size)
//int errcode;
//const regex_t *preg;
//char *errbuf;
//size_t errbuf_size;
{
    register struct rerr *r;
    register size_t len;
    register int target = errcode &~ REG_ITOA;
    register char *s;
    char convbuf[50];
```

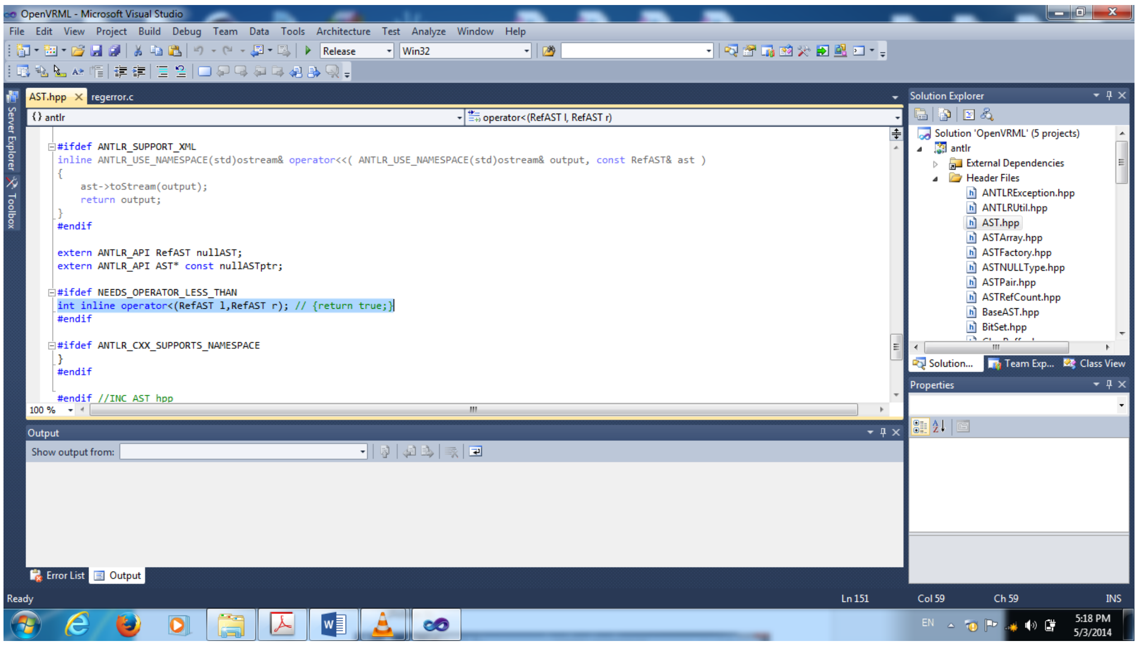
Picture 12. Editing regerror.c by commenting the lines.

The original regex project’s AST.hpp header file:

```
inline operator<(RefAST l,RefAST r); // {return true;}
```

A modified regex project’s AST.hpp header file (Picture 13):

```
int inline operator<(RefAST l,RefAST r); // {return true;}
```



```
#ifdef ANTLR_SUPPORT_XML
inline ANTLR_USE_NAMESPACE(std)ostream& operator<<( ANTLR_USE_NAMESPACE(std)ostream& output, const RefAST& ast )
{
    ast->toStream(output);
    return output;
}
#endif

extern ANTLR_API RefAST nullAST;
extern ANTLR_API AST* const nullASTptr;

#ifdef NEEDS_OPERATOR_LESS_THAN
int inline operator<(RefAST l,RefAST r); // {return true;}
#endif

#ifdef ANTLR_CXX_SUPPORTS_NAMESPACE
}
#endif

#endif //INC_AST_hpp
```

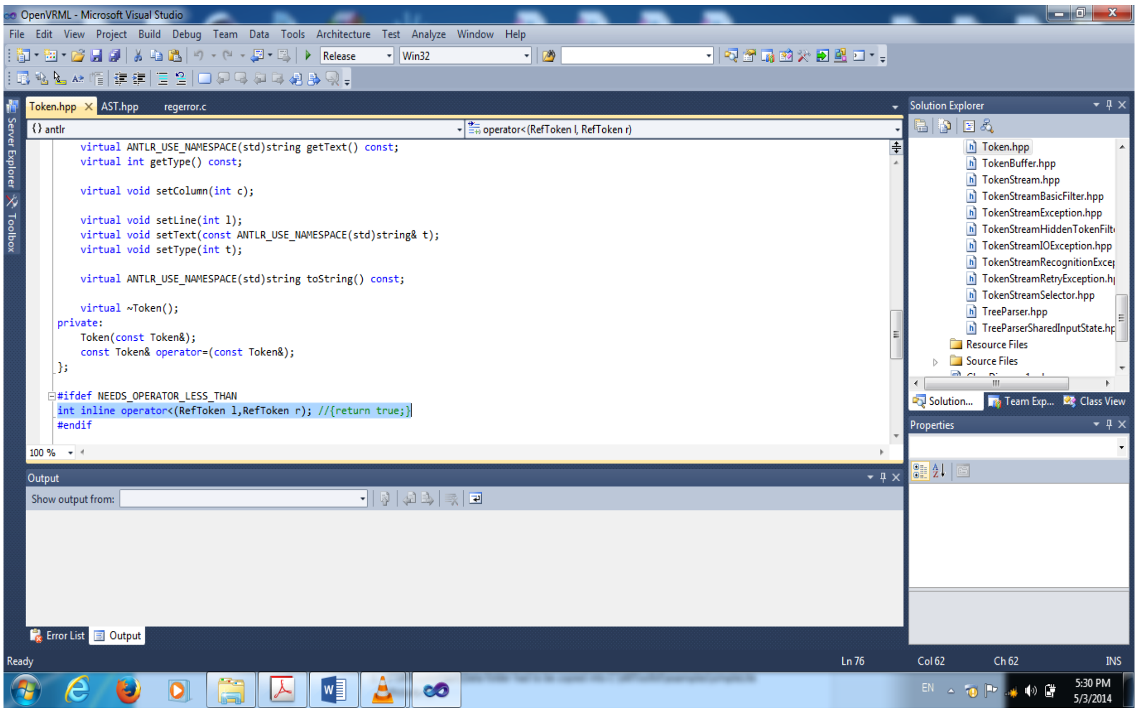
Picture 13. Editing AST.hpp header file.

The original antlr project's Token.hpp header file:

```
inline operator<(RefToken l,RefToken r); //{return true;}
```

A modified antlr project's Token.hpp header file (Picture 14):

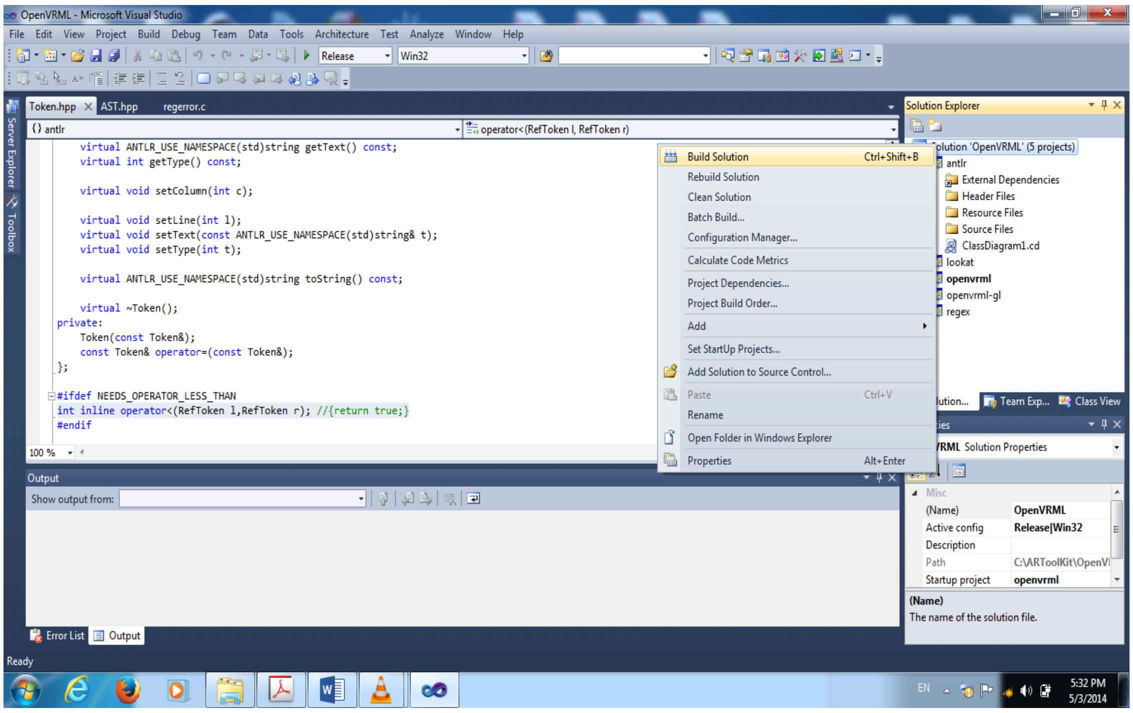
```
int inline operator<(RefToken l,RefToken r); //{return true;}
```



```
    virtual ANTLR_USE_NAMESPACE(std)string getText() const;
    virtual int getType() const;

    virtual void setColumn(int c);

    virtual void setLine(int l);
    virtual void setText(const ANTLR_USE_NAMESPACE(std)string& t);
    virtual void setType(int t);

    virtual ANTLR_USE_NAMESPACE(std)string toString() const;

    virtual ~Token();
private:
    Token(const Token&);
    const Token& operator=(const Token&);
};

#ifdef NEEDS_OPERATOR_LESS_THAN
int inline operator<(RefToken l,RefToken r); //{return true;}
#endif
```

Picture 14. Editing Token.hpp header file.

When the last change has been made, the OpenVRML solution can be build (Picture 15).



```
    virtual ANTLR_USE_NAMESPACE(std)string getText() const;
    virtual int getType() const;

    virtual void setColumn(int c);

    virtual void setLine(int l);
    virtual void setText(const ANTLR_USE_NAMESPACE(std)string& t);
    virtual void setType(int t);

    virtual ANTLR_USE_NAMESPACE(std)string toString() const;

    virtual ~Token();
private:
    Token(const Token&);
    const Token& operator=(const Token&);
};

#ifdef NEEDS_OPERATOR_LESS_THAN
int inline operator<(RefToken l,RefToken r); //{return true;}
#endif
```

(Name)	OpenVRML
Active config	Release|Win32
Description	
Path	C:\ARToolKit\OpenV
Startup project	openvrml

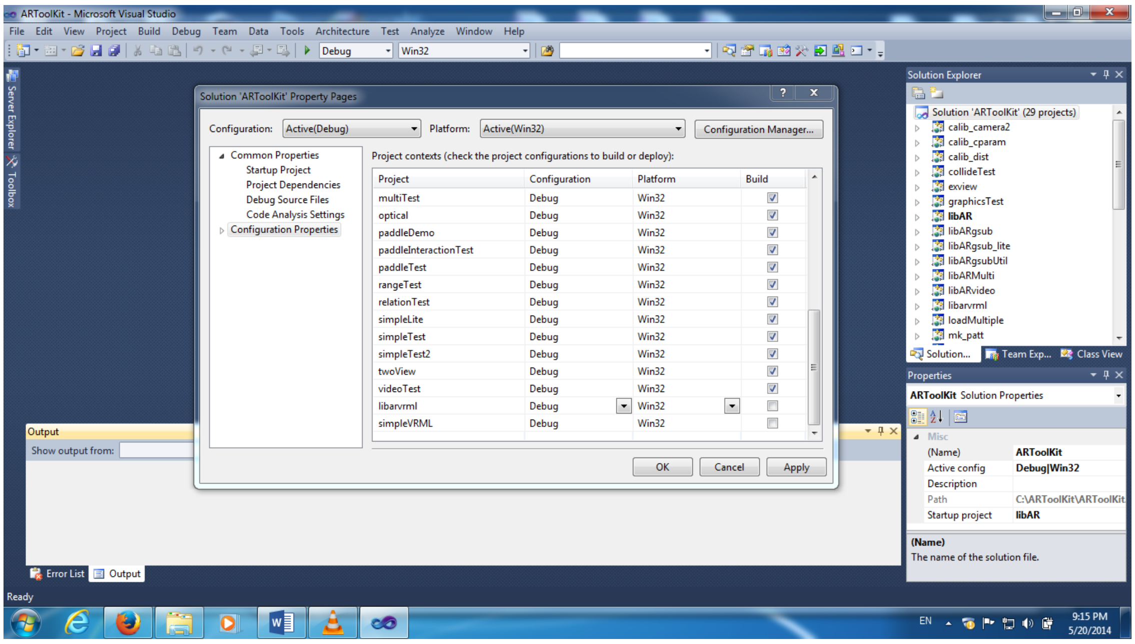
(Name)
The name of the solution file.

Picture 15. Building Solutions.

After projects are built successfully, 4 library files have to be copied into C:\ARToolKit\lib folder. These library files are and can be found in these locations:

- antlr.lib, can be found in C:\ARToolKit\OpenVRML\src\openvrml-0.14.3\ide-projects\Windows\VisualC7\OpenVRML\antlr\Release folder
- openvrml.lib, can be found in C:\ARToolKit\OpenVRML\src\openvrml-0.14.3\ide-projects\Windows\VisualC7\OpenVRML\openvrml\Release folder
- openvrml-gl.lib, can be found in C:\ARToolKit\OpenVRML\src\ openvrml-0.14.3\ide-projects\Windows\VisualC7\OpenVRML\openvrml-gl\Release folder
- regex.lib, can be found in C:\ARToolKit\OpenVRML\src\openvrml-0.14.3\ide-projects\Windows\VisualC7\OpenVRML\regex\Release folder

When all the library files are copied, ARToolKit.sln file is being opened again from the C:\ARToolKit folder with Visual Studio 2010. When Visual Studio opens solutions, the program is changed to Debug mode. Solution ’ARToolKit’ (29 projects) property pages is opened (Picture 16). In this window, Configuration Properties is chosen and unchecked libarvrml and simpleVRML boxes are checked and confirmed by clicking Apply and OK button.



Configuration: Active(Debug) Platform: Active(Win32)

Project contexts (check the project configurations to build or deploy):

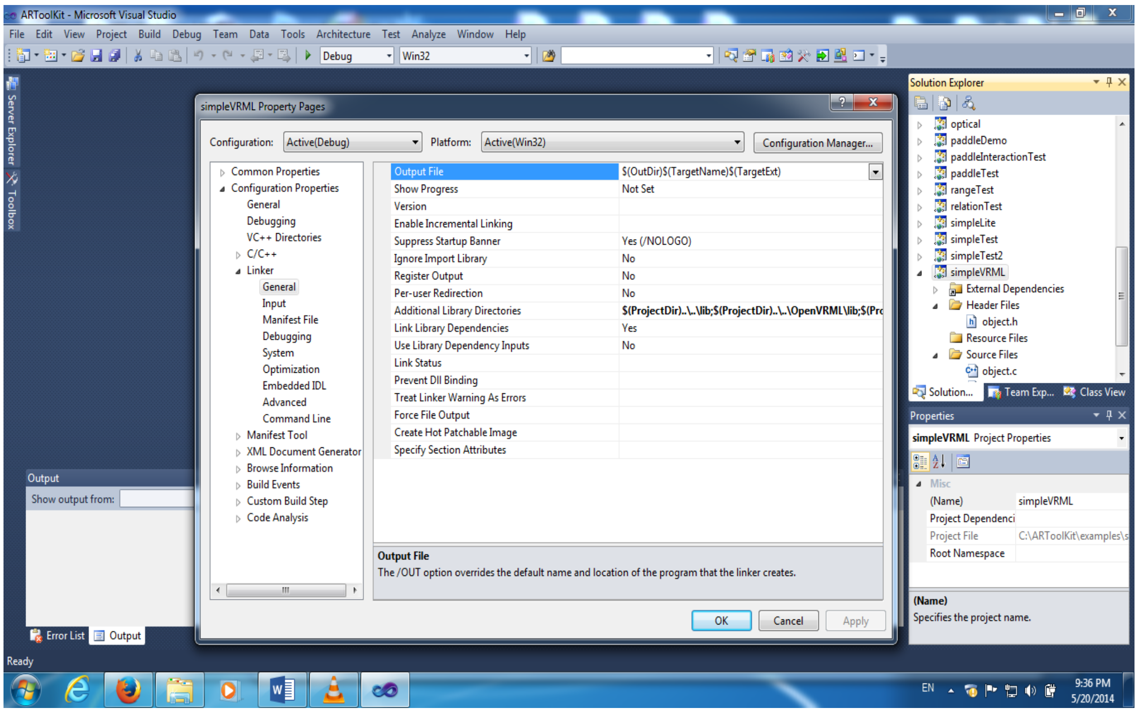
Project	Configuration	Platform	Build
multiTest	Debug	Win32	✓
optical	Debug	Win32	✓
paddleDemo	Debug	Win32	✓
paddleInteractionTest	Debug	Win32	✓
paddleTest	Debug	Win32	✓
rangeTest	Debug	Win32	✓
relationTest	Debug	Win32	✓
simpleLite	Debug	Win32	✓
simpleTest	Debug	Win32	✓
simpleTest2	Debug	Win32	✓
twoView	Debug	Win32	✓
videoTest	Debug	Win32	✓
libarvrml	Debug	Win32	
simpleVRML	Debug	Win32	

(Name)	ARToolKit
Active config	Debug|Win32
Description	
Path	C:\ARToolKit\ARToolKit
Startup project	libAR

(Name)
The name of the solution file.

Picture 16. ARToolKit solution property pages.

Next, simpleVRML properties is opened and property pages appears and Linker (General) is chosen from the Configuration Properties. Output file option is changed to be <inherit from parents or project defaults> (Picture 17). These actions are also made when changing the Visual Studio 2010 to Release mode.



Configuration: Active(Debug) Platform: Active(Win32)

Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	$(ProjectDir)..\..\lib;$(ProjectDir)..\..\OpenVRML\lib;$(Pr
Link Library Dependencies	Yes
Use Library Dependency Inputs	No
Link Status	
Prevent Dll Binding	
Treat Linker Warning As Errors	
Force File Output	
Create Hot Patchable Image	
Specify Section Attributes	

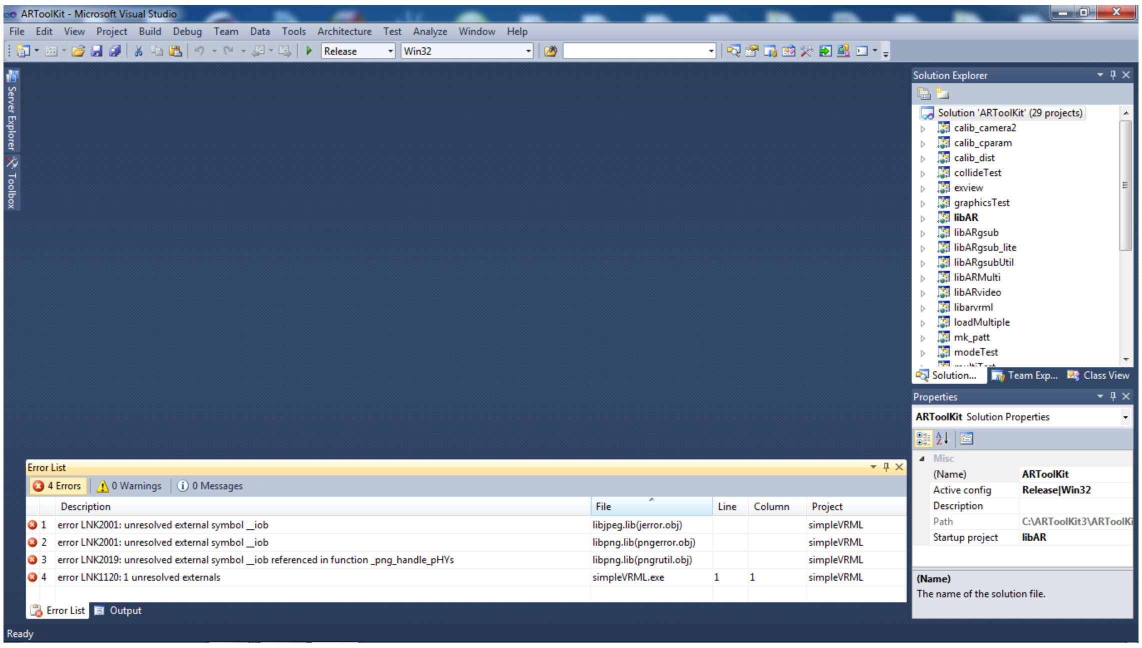
Output File
The /OUT option overrides the default name and location of the program that the linker creates.

(Name)	simpleVRML
Project Dependenci	
Project File	C:\ARToolKit\examples\s
Root Namespace	

(Name)
Specifies the project name.

Picture 17. SimpleVRML property pages.

SimpleVRML project is built in Release mode. When building the code, the project was not built correctly with these configurations. The errors (Picture 18) that were shown in Visual Studio 2010:



Error List — 4 Errors | 0 Warnings | 0 Messages

	Description	File	Line	Column	Project
1	error LNK2001: unresolved external symbol __iob	libjpeg.lib(jerror.obj)			simpleVRML
2	error LNK2001: unresolved external symbol __iob	libpng.lib(pngerror.obj)			simpleVRML
3	error LNK2019: unresolved external symbol __iob referenced in function _png_handle_pHYs	libpng.lib(pngrutil.obj)			simpleVRML
4	error LNK1120: 1 unresolved externals	simpleVRML.exe	1	1	simpleVRML

ARToolKit Solution Properties

Misc	
(Name)	ARToolKit
Active config	Release|Win32
Description	
Path	C:\ARToolKit3\ARToolKi
Startup project	libAR

(Name)
The name of the solution file.

Picture 18. Screenshot of the error list.

These library files are needed:

- libjpeg.lib (popular library file supporting the JPEG image compression)
- libpng.lib (popular library file supporting the PNG image compression)

These files were included into C:\ARToolKit\lib folder. The errors appeared because of the missing library files [29]. After library files were copied into C:\ARToolKit\lib, SimpleVRML project was built again without any errors. SimpleVRML.exe can be found in this folder:

C:\ARToolKit\examples\simpleVRML\Release

10 CALIBRATING THE CAMERA

The parameters should be sufficient for a wide range of different cameras. However using a relatively simple camera calibration technique it is possible to generate a separate parameter file for the specific cameras that are being used. In a video-see through AR interface, if the camera parameters are known then the video image can be warped to remove camera distortions. [30]

There are default camera parameters in the camera_para.dat file which is located in C:\ARToolKit\bin\Data folder. Changes can be made to the camera parameter file, this gives more accurate marker detection. [30] Camera calibration can be made for different type of cameras. For this thesis, camera calibration is made for OKER HD 335 HD Web Cam.

ARToolKit gives two different ways to make calibration [30]:

- Two Step Calibration Approach, which is more accurate.
- One Step Calibration Approach, gives accuracy enough for image overlay.

9.1 Two Step Camera Calibration

The Two step calibration is described here because it is used in this project to get more accurate results. There are pdf files that are used for camera calibration. The files are located in C:\ARToolKit\patterns folder. Two pattern files are calib_cpara.pdf and calib_dist.pdf. These files have to be printed out before starting camera calibration. Printed documents are placed to flat surface. [30]

The calib_cpara.pdf file is a grid pattern of 7 horizontal lines and 9 vertical lines that are exactly 40 mm apart from each other. The calib_dist.pdf file is a pattern that contains 6 x 4 dots which are 40 mm apart from each other. [30]

10.1.1 Calculating the Lens Distortion

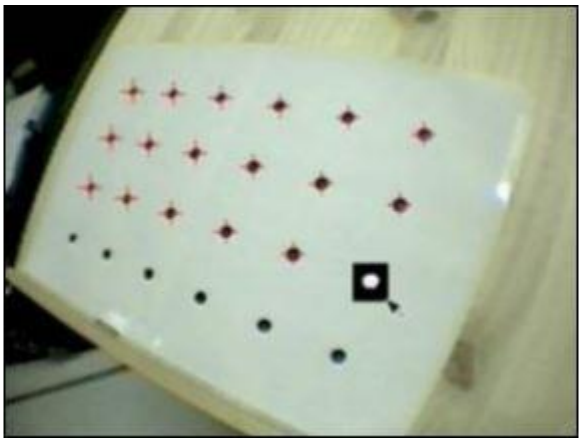
Camera calibration program file can be found in C:\ARToolKit\bin folder. The program calib_dist.exe is used to measure the image center point and lens distortion and calib_param is used to the other camera properties. The program calib_dist is executed first [30]. When the calib_dist file is running, following output can be seen in a terminal window:

```
Camera image size (x,y) = (1280,720)

-----------
Press mouse button to grab first image,
or press right mouse button or [esc] to quit.
```

A window is showing a live video feed. The camera is pointed to calib_dist pattern so that all the dots are in view and are clicked with the left mouse button. When calibrating and pushing the right mouse button, video image freezes. After this, image is being clicked with a left mouse button and dragged to draw a black rectangle over each dot. Each dot are covered with a black drawn rectangle in following order: [30]

```
1   2  3  4  5  6
7   8  9 10 11 12
13 14 15 16 17 18
19 20 21 22 23 24
```

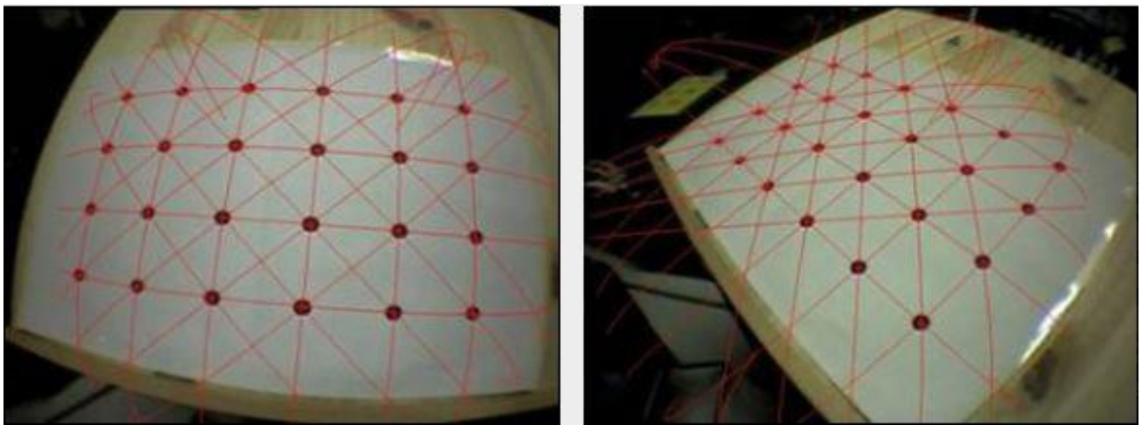
Each rectangle that is drawn, software finds the dot enclosed by the rectangle and places a red cross at its center (Picture 19). When all the 24 dots are found and the left mouse button is being pushed again, the program stores the position of the dots and unfreezes the video image. This image prosessing action can be proceeded 5-10 times. [30] In this thesis, 10 images are being processed from various angles and points to make calibration and marker detection more accurate.



Picture 19. Example of marking calibration dots.

After 10 images have been processed, the image capture is being stopped by clicking the right mouse button and calib_dist code generates the final key values. These values are the center x and y values and distortion factor. The generated values are different for every camera. The values for the OKER HD 335 HD Web Cam are written into calib_cparam file.

Correct parameters can be checked when left mouse button is being clicked. First grabbed image is shown with red lines drawn through the calibration dots (Picture 20). When calibration is done, red lines are passed through the center of the each dot. All taken images and calibration can be checked each time when the left mouse button is being clicked. The program calib_dist.exe program stops when the right mouse button is being clicked. [30] Calib_dist program has now been completed.



Picture 20. Two examples of calibrated image results.

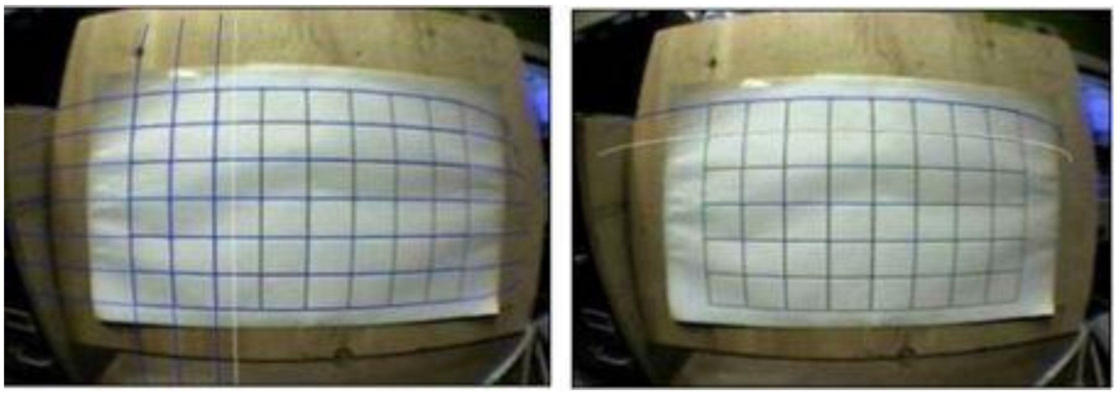
10.1.2 Calibrating Lens and Other Parameters

The camera calibration program for the lens and other parameters program file can be found in C:\ARToolKit\bin folder. The program calib_cparam.exe is executed in the command prompt window. A live video stream window will appear. The calibration board is placed in front of the OKER HD 335 HD Web Cam. The board is placed in the front of the camera so all the grid lines are visible as large as possible. [30]

The image is being grabbed and the left mouse button is being clicked. A white horizontal line appears overlaid on the image. The white line can be now moved to the top of the black grid line. The white line is placed on the top of the black grid line as close as possible. The line is being moved using the up and down arrow keys. When the line is being rotated clockwise and anticlockwise, the left and right arrow keys are clicked to move the line. Enter key is being pushed. The white line changes the color to blue and another white line appears. When all the horizontal lines are tagged, the vertical line appears. [30]

Same process are made to the horizontal and vertical lines. The first vertical line is placed over the grid line that is the most left one. This procedure is being proceeded until every grid line from left to right has being placed. All horizontal lines are now tagged and a vertical line appears. [30]

The placement of these lines are very important and have to be placed in this particular order. Now, all the 16 lines are drawn on the screen. When the first image has been processed, the grid pattern is moved 100 mm away from the camera and the same process is made again. The capture of the line placement can be seen in the Picture 21. The placement process is being repeated 5 times and the calibration pattern is now 500 mm away from the camera. When finalizing the last calibration step, the program will automatically calculate the camera parameters. The command prompt window shows camera parameters which are stored in oker_parameters.dat file. [30]



Picture 21. Examples of horizontal and vertical line placement.

To use the new camera parameters, the oker_parameters.dat is renamed to camera_para.dat and the file is being placed into C:\ARToolKit\bin\Data folder. The new values are also fixed in the source code calib_cparam.c in the C:\ARToolKit\util\calib_cparam\ directory. The file can be used immediately in the ARToolKit sample programs. [30] Camera calibration is used to get better tracking results.

The distance between grid line values of the calib_cparam_sub.c source code file can be modified if needed. When modifying the code, these lines should be changed [30]:

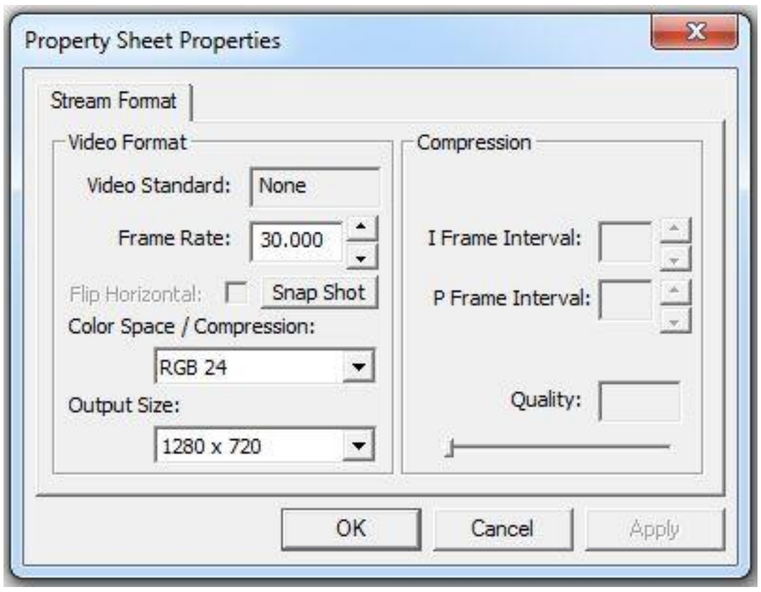
```
inter_coord[k][j][i+7][0] = 40.0*i;
inter_coord[k][j][i+7][1] = 40.0*j;
inter_coord[k][j][i+7][2] = 100.0*k;
```

In the code, 40.0 is the current distance of the grid lines and the 100.0 is the distance that the pattern should be moved back from the camera each time. Measurement times can also be changed. If the code needs to be changed, following line in the below needs to be found from source code. The number 5 stands for how many times calibration distance is being changed [30].

```
.*loop_num = 5;
```

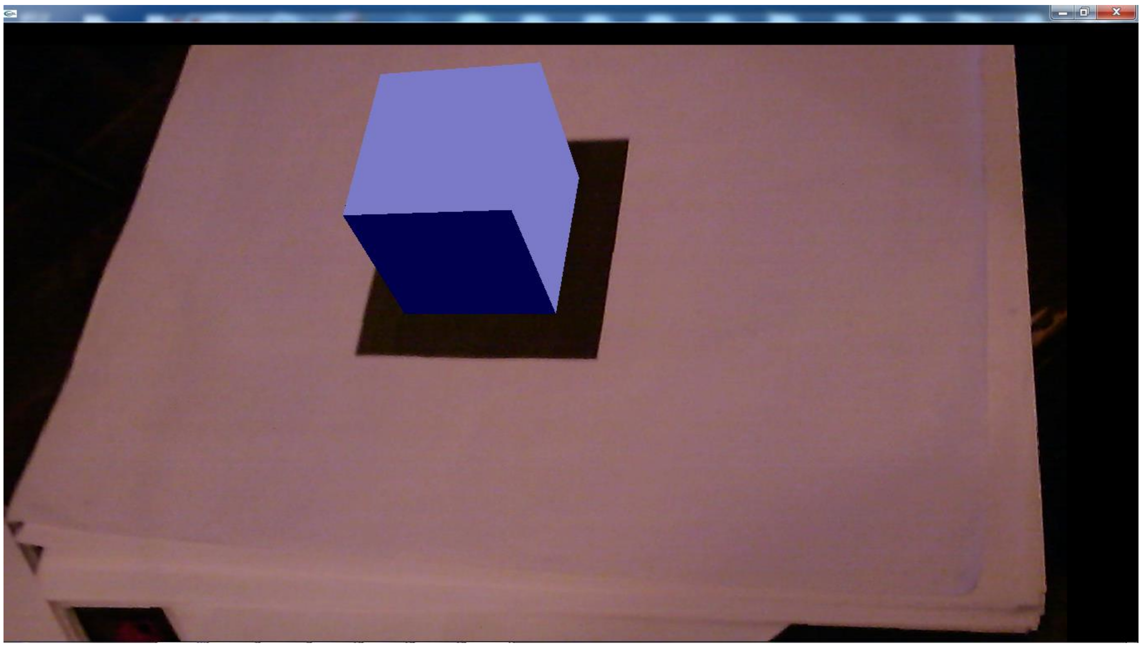
11 TESTING ARTOOLKIT EXAMPLES

ARToolKit programs can now be tested. All the built project files can be found in C:\ARToolKit\examples folder. The simpleTest.exe program is executed. It is located in C:\ARToolKit\bin folder [31]. An example pattern hiroPatt.pdf is used when SimpleTest.exe is opened from Windows Explorer window. Testing shows that simpleTest program works both in a Command Prompt window and when ran with Microsoft Visual Studio 2010. The program shows simple camera stream window. Before live video streaming window is shown, different options can be selected. The camera is tested with these configurations (Picture 22):


Property Sheet Properties
Stream Format
Video Format
Video Standard: None
Frame Rate: 30.000
Flip Horizontal:
Color Space / Compression: RGB 24
Output Size: 1280 x 720
Compression
I Frame Interval:
P Frame Interval:
Quality:

Picture 22. Property sheet properties.

Interaction of the camera and the Hiro marker shows that simpleTest.exe program works successfully. A screenshot is taken and there can be seen a blue virtual block aligned on the Hiro marker (Picture 23).



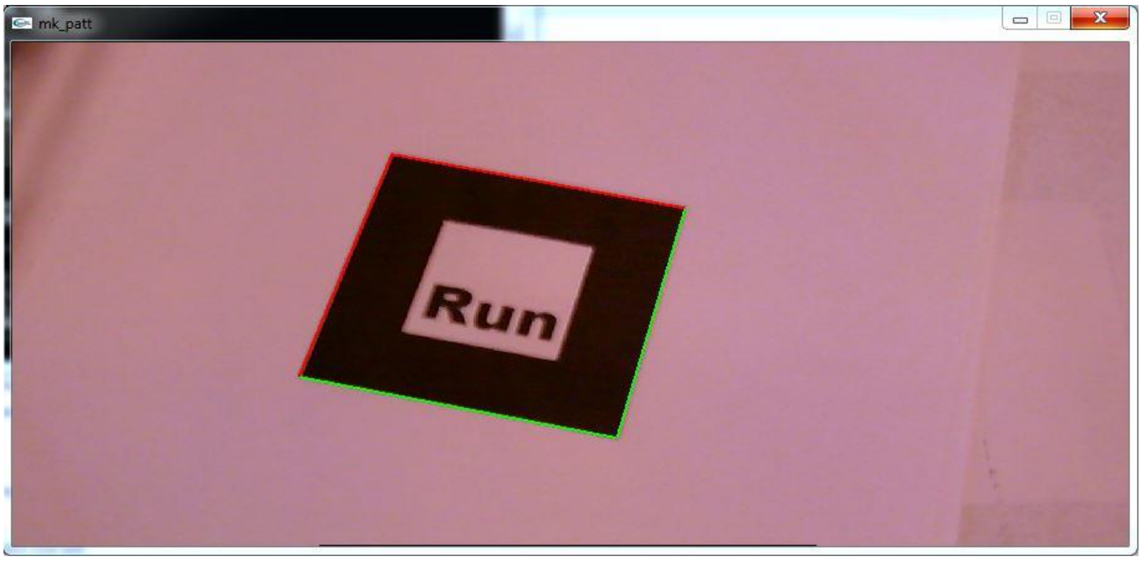
Picture 23. Screenshot of the blue virtual block aligned on the Hiro marker.

12 MAKING OWN AUGMENTED REALITY

Testing shows that OKER HD 335 HD Web Cam sees the Hero marker. Next phase is to create animated 3D characters. Markers that are used have text inside of the each marker. Markers are called Walk, Run, Jump and Crouch. These markers are introduced earlier in the template markers section. To use the new markers in ARToolKit, each marker needs to be teached to the camera. When the camera sees markers, animated 3D characters supposed to appear and do different movement on each marker.

12.1 Teaching Markers for ARToolKit

To detect new markers, these have be teached to the camera. A program for teaching markers is located in C:\ARToolKit\bin folder. The program mk_patt.exe is executed in a command promt window. The camera needs to be rotated until the red corner of the highlighted square is the top left hand corner of the square in the video image (Picture 24). [32]



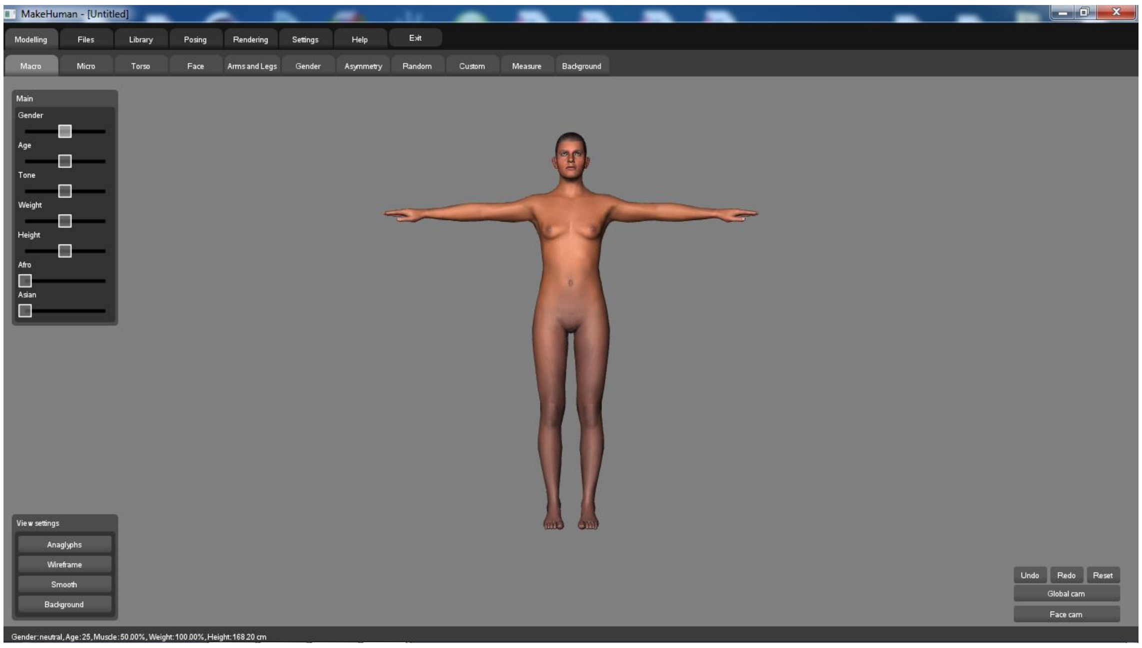
Picture 24. Screenshot of teaching the Run marker.

When the marker is saved, it is copied into Data folder. The Run marker can now be used with SimpleVRML program.

12.2 Creating a Charater with Makehuman

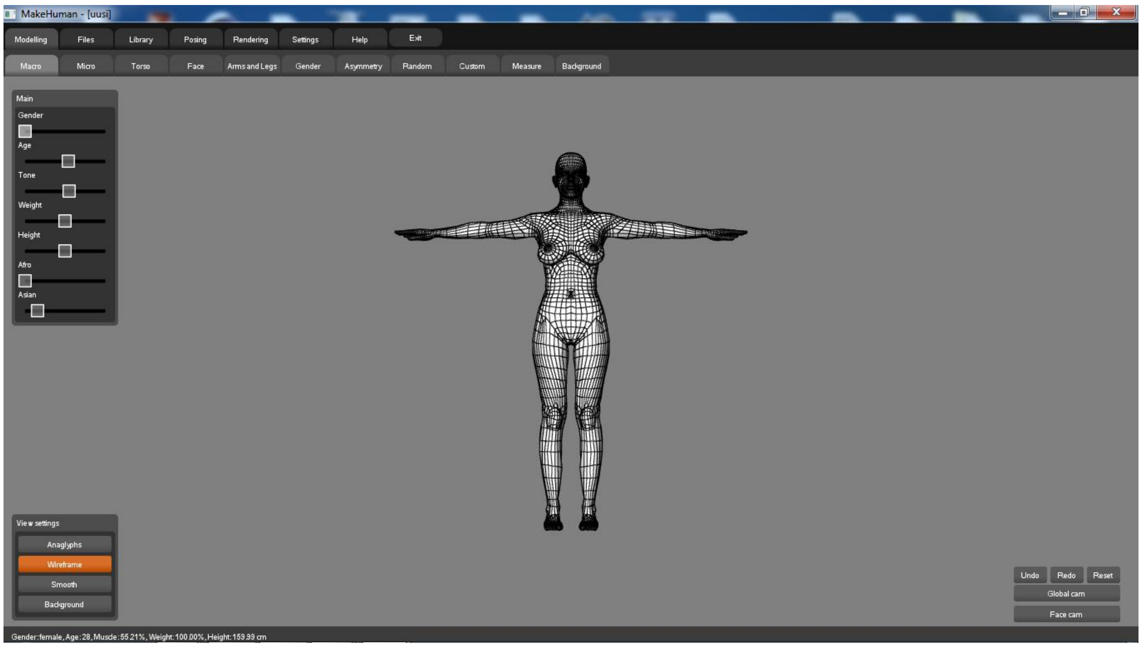
Makehuman 1.0 alpha 7 is being used to create custom made 3D human character. The program is really easy to use and its Graphic User Interface (GUI) makes 3D human modelling fast and simple. When a character has been made, it can be easily used with many modelling and rendering programs to create and export human figures. These 3D figures can be processed again with other programs that can handle 3D images and animations.

Makehuman starts with a default window that shows a human figure facing you with outstretched arms. In the window, there can be also seen multiple toolbars and toolbox controls (Picture 27). Randomly chosen configurations are made for this project. After configurations, the aim was to create 3D female model (Picture 28).



Gender:neutral, Age:25, Muscle:50.00%, Weight:100.00%, Height:168.20 cm

Picture 25. Defaut window of Makehuman 1.0 alpha 7.



Gender:female, Age:28, Muscle:55.21%, Weight:100.00%, Height:159.99 cm

Picture 26. Created 3D female figure in wireframe format.

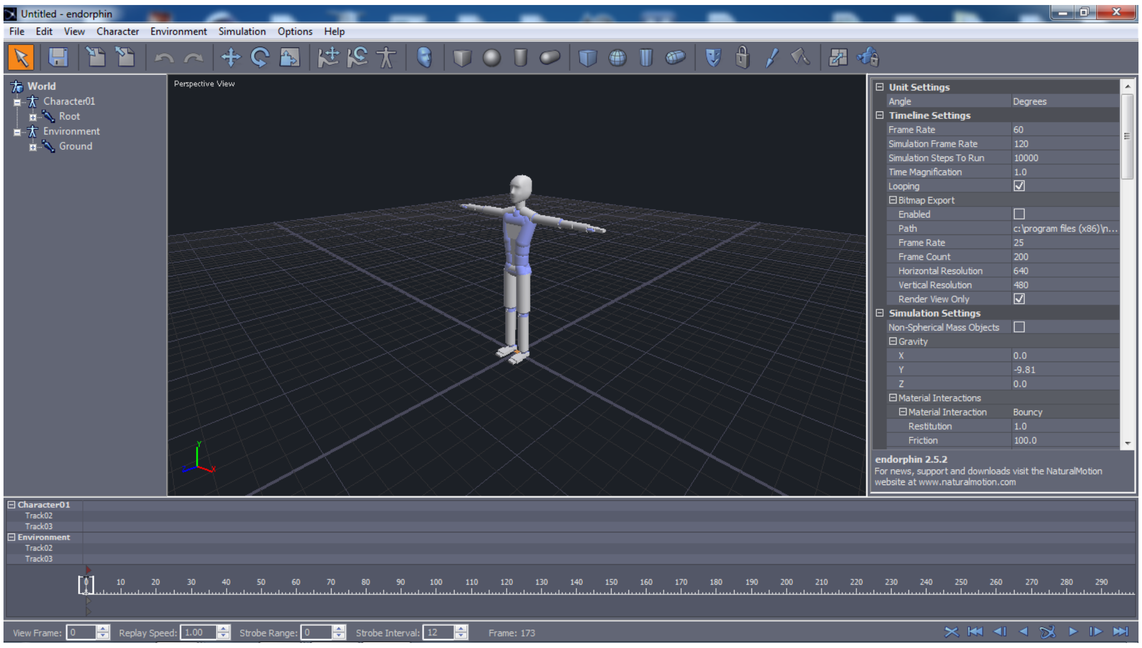
After character creation, it is exported as an object file.

12.3 Making Joints and Animations with Endorphin 2.5.2

Endorphin is 3D animation software that is based on Dynamic Motion Synthesis (DMS). Endorphin is being used to create Biovision Hierarchy (BVH) file and animations. The BVH format is developed by BioVision. It was created to give motion capture data to the people. [33] When starting Endorphin software, a default window shows 3D puppet on the screen (Picture 27). The pre-made animation imports are used to make movements for the puppet. The Picture 28 shows import settings. Created BVH movements are:

- Run
- Walk
- Jump
- Crouch

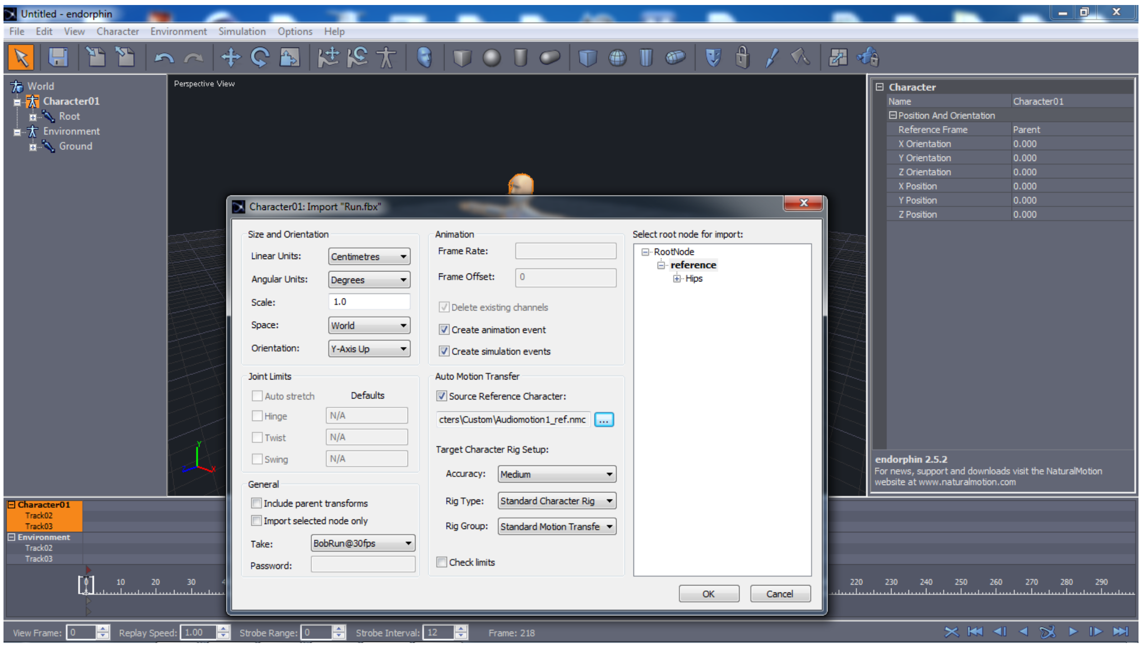
Every movement is done by importing pre-made animations. There are now 4 BVH skeletons with animations to each marker.



Unit Settings	
Angle	Degrees
Timeline Settings	
Frame Rate	60
Simulation Frame Rate	120
Simulation Steps To Run	10000
Time Magnification	1.0
Looping	☑
Bitmap Export	
Enabled	☐
Path	c:\program files (x86)\n...
Frame Rate	25
Frame Count	200
Horizontal Resolution	640
Vertical Resolution	480
Render View Only	☑
Simulation Settings	
Non-Spherical Mass Objects	☐
Gravity	
X	0.0
Y	-9.81
Z	0.0
Material Interactions	
Material Interaction	Bouncy
Restitution	1.0
Friction	100.0

endorphin 2.5.2
For news, support and downloads visit the NaturalMotion website at www.naturalmotion.com

Picture 27. Default screen of Endorphin software.



Character01: Import "Run.fbx"

Size and Orientation
Linear Units: Centimetres
Angular Units: Degrees
Scale: 1.0
Space: World
Orientation: Y-Axis Up

Joint Limits
☐ Auto stretch — Defaults
☐ Hinge N/A
☐ Twist N/A
☐ Swing N/A

General
☐ Include parent transforms
☐ Import selected node only
Take: BobRun@30fps
Password:

Animation
Frame Rate:
Frame Offset: 0
☑ Delete existing channels
☑ Create animation event
☑ Create simulation events

Auto Motion Transfer
☑ Source Reference Character:
cters\Custom\Audiomotion1_ref.nmc
Target Character Rig Setup:
Accuracy: Medium
Rig Type: Standard Character Rig
Rig Group: Standard Motion Transfe
☐ Check limits

Select root node for import:
RootNode
reference
Hips

Character	
Name	Character01
Position And Orientation	
Reference Frame	Parent
X Orientation	0.000
Y Orientation	0.000
Z Orientation	0.000
X Position	0.000
Y Position	0.000
Z Position	0.000

endorphin 2.5.2
For news, support and downloads visit the NaturalMotion website at www.naturalmotion.com

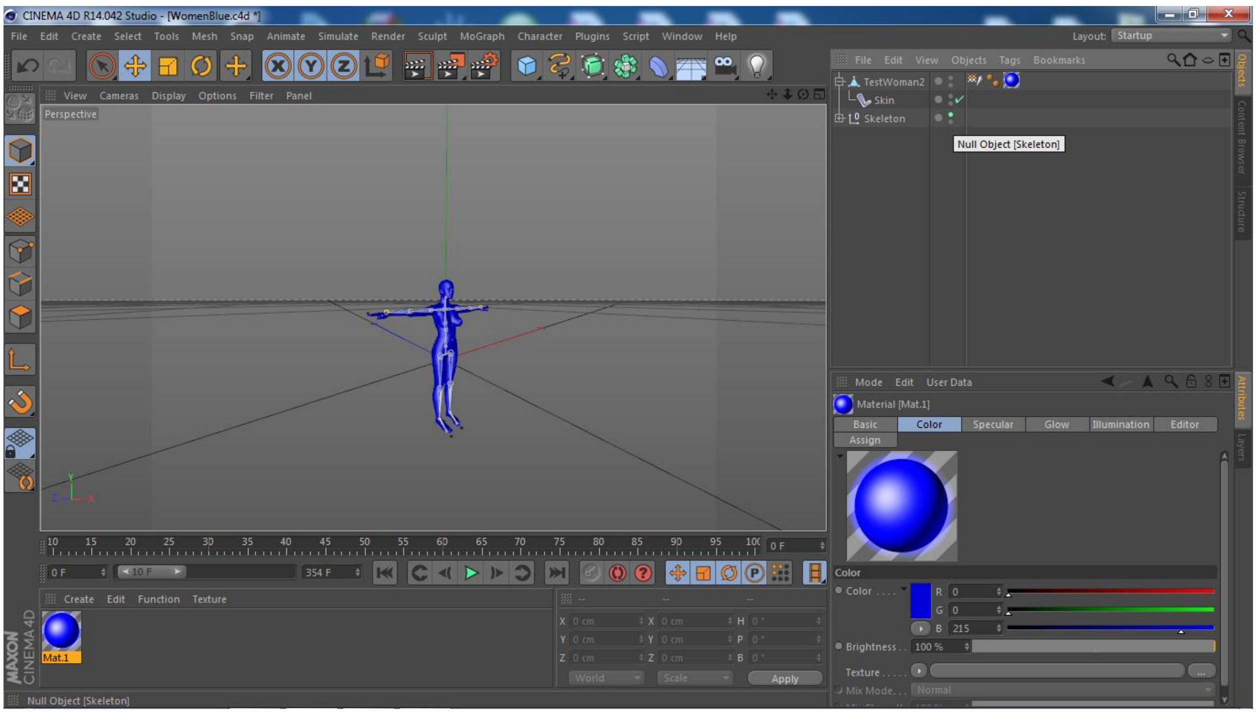
Picture 28. Importing BVH file for the run movement.

12.4 Connecting the Object and Joints with Cinema 4D

Cinema 4D is a 3D modelling program to create 3D objects and animations. For this thesis, the created BVH joints and 3D character are being combined using Cinema 4D. Female character OBJ files are imported to Cinema 4D without textures. The 3D object is being painted with randomly chosen blue color. After this, joints are placed and fitted inside of the 3D female character (Picture 29). The characters and joints are combined and the model is now ready to be used in ARToolKit. All the animated character files worked correctly in Cinema 4D. The files of the animated characters are exported as VRML files into this folder:

C:\ARToolKit\examples\simpleVRML\Release\Wrl

VRML files are used because ARToolKit only supports VRML and X3D files.


CINEMA 4D R14.042 Studio - [WomenBlue.c4d *]
File Edit Create Select Tools Mesh Snap Animate Simulate Render Sculpt MoGraph Character Plugins Script Window Help
Layout: Startup
View Cameras Display Options Filter Panel
Perspective
TestWoman2
Skin
Skeleton
Null Object [Skeleton]
Mode Edit User Data
Material [Mat.1]
Basic Color Specular Glow Illumination Editor
Assign
Color
R 0
G 0
B 215
Brightness 100 %
Texture
Mix Mode Normal
Create Edit Function Texture
Mat.1
World Scale Apply
Null Object [Skeleton]

Picture 29. The 3D model with combined joints.

12.5 Testing VRML Files with ARToolKit

The 4 character files can be now tested. Before testing, DSVL.dll, DSVLd.dll , glut32.dll, js32.dll , libARvideo.dll and libARvideod.dll are copied from C:\ARToolKit\bin directory into this folder:

C:\ARToolKit\examples\simpleVRML\Release\

Also Data and Wrl folders are copied into same directory. There are also need for new DAT files for every exported character. These files are created to the same directory as VRML files are located in. For example, created DAT file for running character is:

```
WomanRun.wrl
0.0 0.0 0.0                        # Translation
120.0 90.0 90.0 90.0               # Rotation
5.0 5.0 5.0                        # Scale
```

Values for Data\object_data_vrml.dat file needs to be corrected. The inserted values for the running character are:

```
#pattern 3
VRML     Wrl/WomanRun.dat
Data/patt.womanrun
80.0
0.0 0.0
```

Testing can now be started by using patt.run, patt.walk, patt.crouch and patt.jump markers. When OKER HD 335 HD Web Cam is being pointed to different markers, the 3D character appears on the top of each marker. Testing shows that animations are not shown because Cinema 4D does not fully support VRML scene exportation. Testing only shows that the camera sees markers.

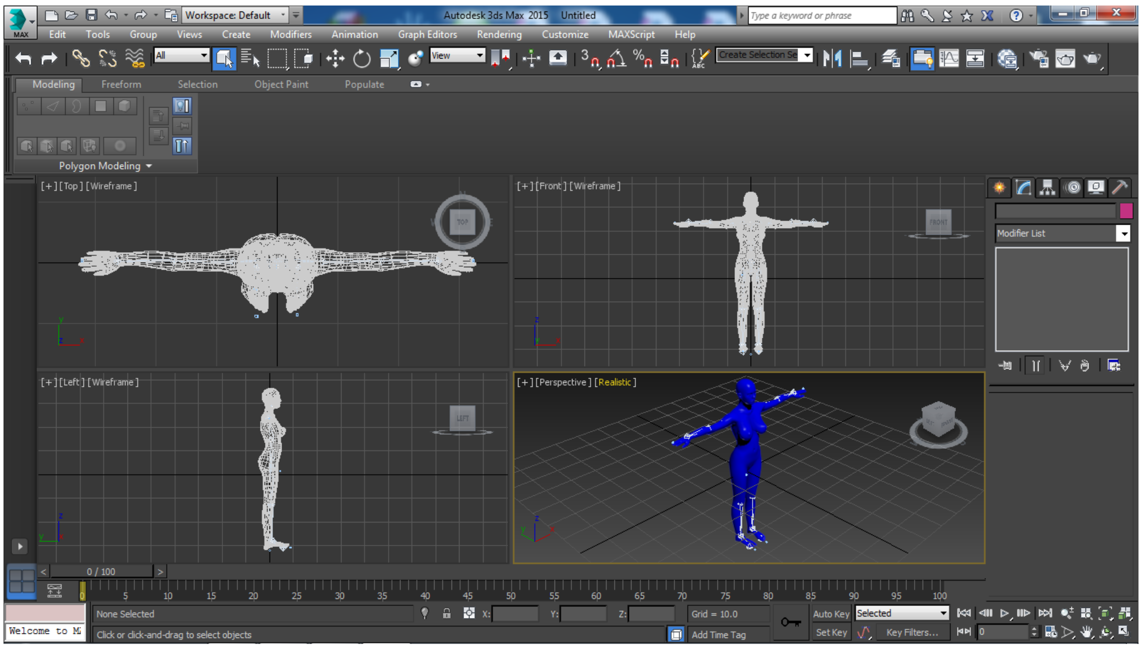
12.6 Creating VRML Scenes with Autodesk 3ds Max 2015

Autodesk 3ds Max is a 3D modelling software. It is used for 3D modelling and generating animated scenes. The program supports more different file formats than Cinema 4D. That is why the animated characters are exported as an FBX file by using Cinema 4D as an exporter.

FBX format is a commonly used in Autodesk products. It was originally developed by Kaydara for Motion Builder and owned by Autodesk since 2006. It is one of the main 3D exchange formats used by different 3D tools and programs. [34]

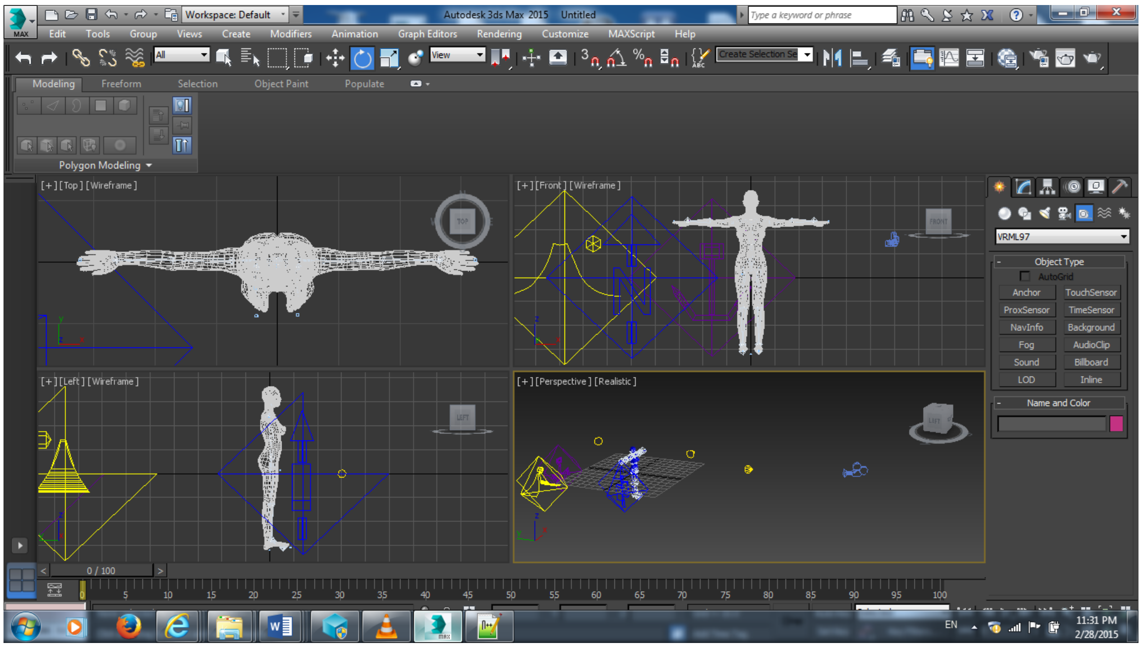
The created 3D characters in FBX format are imported to Autodesk 3ds Max 2015 in FBX format (Picture 30). When testing animations, these worked perfectly in Autodesk. There are certain configurations that have to be fulfilled to build working VRML scenes in Autodesk [35]. These prerequisites are added to VRML scene (Picture 31):

- Background
- Lights
- NavInfo
- Anchor



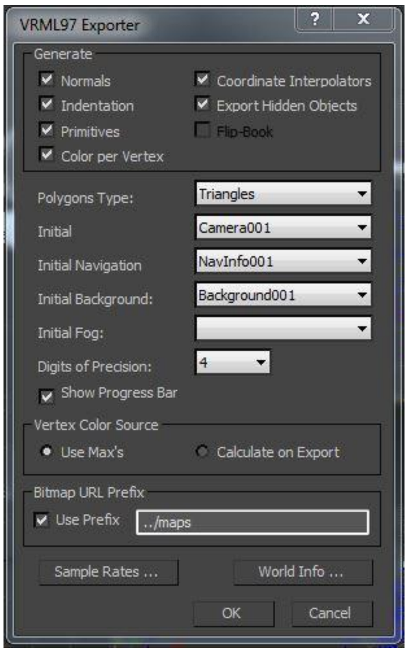
Picture 30. Imported file in FBX format.

All the prerequisites are randomly placed into VRML scene (Picture 31).



Picture 31. FBX file with needed prerequisites.

FBX files can now be exported to VRML format. Picture 32 shows used VRML configurations.



VRML97 Exporter

Generate
- [✓] Normals
- [✓] Coordinate Interpolators
- [✓] Indentation
- [✓] Export Hidden Objects
- [✓] Primitives
- [ ] Flip-Book
- [✓] Color per Vertex

Polygons Type:	Triangles
Initial	Camera001
Initial Navigation	NavInfo001
Initial Background:	Background001
Initial Fog:	
Digits of Precision:	4

- [✓] Show Progress Bar

Vertex Color Source
- ● Use Max's
- ○ Calculate on Export

Bitmap URL Prefix
- [✓] Use Prefix ../maps

Sample Rates ... World Info ... OK Cancel

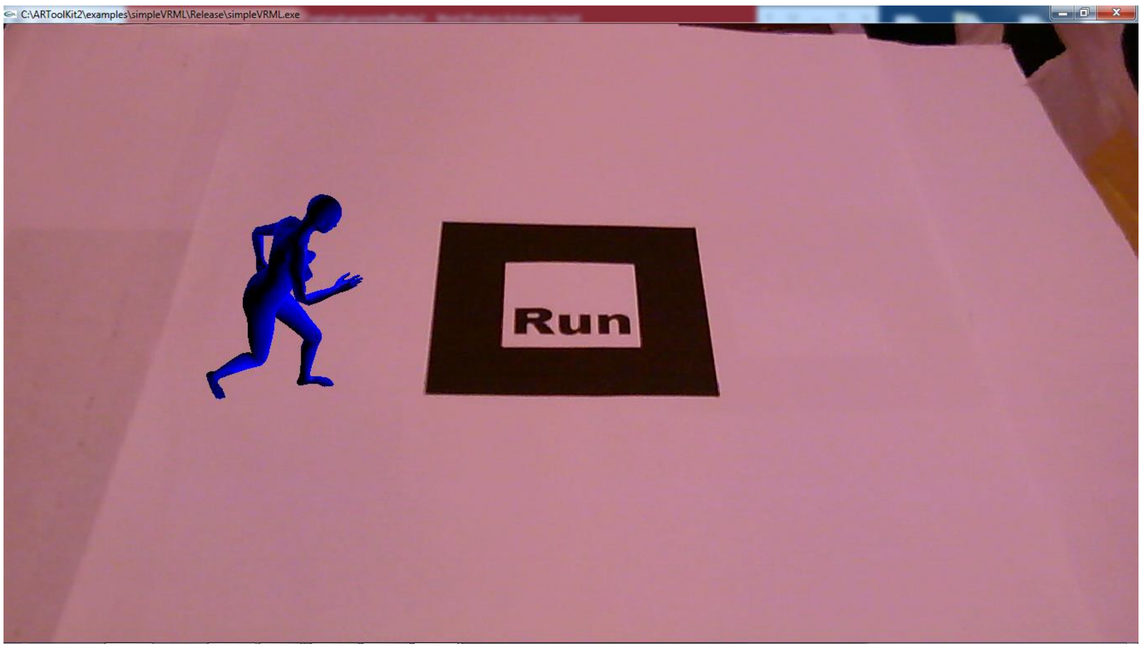
Picture 32. VRML expoter configurations.

When this is done, all the VRML files are copied into this folder:

C:\ARToolKit\examples\simpleVRML\Release\Wrl

12.7 Testing Autodesk VRML Files

The first marker that is being tested is the Run marker. Testing shows that the camera identifies the Run marker and the animated 3D character appears and runs on top of the marker (Picture 33). Everything works like it supposed to. Every marker is being tested and showed to the camera.



Picture 33. Screenshot of the running 3D character.

VRML files that are made by using Autodesk 3DS Max 2015 worked as planned. Several testing shows that the camera identifies each marker and different animated 3D character is displayed on the top of each marker. With these final experiments, it can now be noted that ARToolKit code works when creating augmented reality. The testing also shows that the animated VRML scenes which are done by using Autodesk 3ds Max 2015 are displayed correctly and the files are compatible with ARToolKit.

13 SUMMARY

When creating augmented reality, one needs to have certain knowledge of basic programming skills, and knowing how to compile codes. Equiment that are needed is a computer and simple web camera for detecting markers. ARToolKit gives simple tools to track markers. ARToolKit site also gives basic knowledge of marker detection. There can be found a great amount of documented information about augmented reality and ARToolKit when creating and developing different AR projects. Althougth ARToolKit's source code is from 1999, the testing shows that the code works like it supposed to. Basically, ARToolKit gives an easy way to approach augmented reality.

When prepairing to install ARToolKit, one needs to know that installation and configurating process takes time. Camera calibration is also very time consuming if the 2 step camera calibration is being used.

Modelling skills with 3D software are also needed. Modelling skills that are required depends on developpers skills. In this thesis, 3D modelling was not a problem because of the former 3D modelling proficiency. Only thing that made problems was the firstly chosen 3D modelling software because it did not fully support VRML scene creation. That is why Autodesk 3ds Max 2015 is recommended when creating VRML scenes to ARToolKit.

This Bachelor's thesis was completed succefully by using various program. Completing this project has given knowledge, information, ideas and new skills when it comes to making of augmented reality. The project has also proven that ARToolKit software library gives common tools for developers to start to create and build AR scenes. In this Bachelor's thesis, the results were animated 3D characters which were associating with different markers. The different markers were detected correctly and animated scenes work appropriately.

REFERENCES

[1] S. K. Ong and A. Y. C. Nee, *Virtual and Augmented Reality Applications in Manufacturing*, Springer-Verlag London, 387p.

[2] Kato, H., Billinghurst, M. (Oct, 1999). Marker tracking and hmd calibration for a video-based augmented reality conferencing system, *In Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR 99)*. Retrieved from http://www.hitl.washington.edu/artoolkit/publications.

[3] L. Frank Baum. Envisions Augmented Reality specs in 1901. (Mote Beam 10 September 2012). Retrived from http://moteandbeam.net/the-master-key-l-frank-baum-envisions-ar-glasses-in-1901.

[4] IFI CLAIMS Patent Services. (n. d.). Sensora simulator patent by Helig, Morton, L. Retrieved from http://www.google.com/patents?q=3050870.

[5] Sutherland, Ivan, E. A Head-Mounted Three-Dimensional Display [PDF document]. Retrieved from http://90.146.8.18/en/archiv_files/19902/E1990b_123.pdf.

[6] Mann, Steve. (2012 Nov 2). Eye Am a Camera: Surveillance and Sousveillance in the Glassage Retrieved from http://techland.time.com/2012/11/02/eye-am-a-camera-surveillance-and-sousveillance-in-the-glassage.

[7] Google Glass Project. (Feb 21, 2014). Retrieved from http://www.etceter.com/c-news/p-google-glasses-project.

[8] Lanier, J., Minsky, M., Fisher, S. And Druin, A. (1989 ACM Siggraph Panel Proceedings). Virtual Environments And Interactivity: Windows To The Future. Retrieved from http://www.jaronlanier.com/general.html.

[9] Kangdon, Lee. (March/April 2012). Augmented Reality in

Education and Training [PDF document]. Retrieved from http://www2.potsdam.edu/betrusak/566/Augmented%20Reality%20in%20Education.pdf.

[10] Sung, Dan. (March 1 2012). The history of augmented reality. Retrieved from http://www.pocket-lint.com/news/108888-the-history-of-augmented-reality.

[11] www.cs.unc.edu/~raskar/Office/IWAR_SAR.pdf

[12] Miller, Claire, Cain. The New York Times, Google Searches for Style. (Feb 20 2013). Retrieved from http://www.nytimes.com/2013/02/21/technology/google-looks-to-make-its-computer-glasses-stylish.html?pagewanted=all&_r=1.

[13] Microsoft. (Jan 21, 2015). Microsoft HoloLens [Video file]. Retrieved from https://www.youtube.com/watch?v=aThCr0PsyuA.

[14] ARToolworks, Inc., Seattle, WA, USA.ARToolKit. (n. d.). Retrieved from http://www.hitl.washington.edu/artoolkit.

[15] ARToolworks, Inc., Seattle, WA, USA. (n. d.). ARToolKit, Setting up ArtoolKit. Retrieved from http://www.hitl.washington.edu/artoolkit/documentation/usersetup.htm.

[16] Silicon Graphics International Corp. (n. d.). OpenGL Overview. Retrieved from https://www.opengl.org/about.

[17] Silicon Graphics International Corp. (n. d.). GLUT, The OpenGL Utility Toolkit. Retrieved from https://www.opengl.org/resources/libraries/glut.

[18] Microsoft. (n. d.). Introduction to DirectShow. Retrieved from https://msdn.microsoft.com/en-us/library/windows/desktop/dd390351(v=vs.85).aspx.

[19] TechTerms.com. (n. d.) DirectX. Retrieved from http://techterms.com/definition/directx.

[20] ARToolworks, Inc., Seattle, WA, USA. (n. d.). ARToolKit, How does ARToolKit work?. Retrieved from http://www.hitl.washington.edu/artoolkit/documentation/userarwork.htm.

[21] Hartley, R.I. & Zisserman, A. Multiple View Geometry in Computer Vision. Cambridge University Press, 2000

[22] Siltanen, Sanni. (2012). Theory and applications of marker-based augmented reality [PDF] document]. Retrieved from http://www2.vtt.fi/inf/pdf/science/2012/S3.pdf.

[23] Gonzalez, R.C. & Woods, R.E. Digital image processing. Boston, MA, USA: Addison-Wesley Longman, 2001.

[24] Szeliski, R. Computer Vision: Algorithms and Applications. New York, NY, USA: Springer-Verlag, 2010. Retrieved from http://research.microsoft.com/en-us/um/people/szeliski/book/drafts/szelski_20080330am_draft.pdf [4 January 2012].

[25] Uematsu, Y. & Saito, H. (2007). Improvement of accuracy for 2D marker-based tracking using particle filter. Proceedings of the 17th International Conference on Artificial Reality and Telexistence (ICAT). Washington, DC, USA: IEEE Computer Society. Pp. 183–189.

[26] Li, S. & Xu, C. Efficient lookup table based camera pose estimation for augmented reality. Computer Animation and Virtual Worlds 2011, Vol. 22, No. 1, pp. 47–58. Retrieved from http://dx.doi.org/10.1002/cav.385.

[27] Freeman, R.M., Juliet, S.J. & Steed, A.J. A method for predicting marker tracking error. Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR). Nara-Ken New Public Hall, Nara, Japan, 13–16 Nov. 2007. Pp. 157–160.

[28] Kato, H. Inside ARToolKit [PDF document]. (Oct 1 2010). Retrieved from http://www.hitl.washington.edu/artoolkit/Papers/ART02-Tutorial.pdf.

[29] Andersson, Maxwell. (Aug 31 2009). Configurating ARToolKit [Web log comment]. Retrieved from http://maxwellanderson.com.br/blog/visual-studio/configurando-o-artoolkit-e-a-openvrml-no-microsoft-visual-studio-2008-parte-22/

[30] ARToolworks, Inc., Seattle, WA, USA. (n. d.). ARToolKit, Configurating camera. (n. d.). Retrieved from http://www.hitl.washington.edu/artoolkit/documentation/usercalibration.htm.

[31] ARToolworks, Inc., Seattle, WA, USA. (n. d.). ARToolKit, Example program. Retrieved from http://www.hitl.washington.edu/artoolkit/documentation/userstartup.htm.

[32] ARToolworks, Inc., Seattle, WA, USA. (n. d.). ARToolKit, Developing application part 2. Retrieved from http://www.hitl.washington.edu/artoolkit/documentation/devmulti.htm.

[33] University of Wisconsin: Department of Computer Sciences. (n. d.). BVH Information. Retrieved from http://research.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html.

[34] Blender developer blog by ”ton”. (Aug 10, 2013). FBX binary file format specification [Web log comment]. Retrieved from http://code.blender.org/index.php/2013/08/fbx-binary-file-format-specification.

[35] Neill, H. (n. d.). 3D modelling for Virtual Reality. Retrieved from http://www.tech.plymouth.ac.uk/dmme/cad/pdf/VRML_3dsmax/VR_room_tutorial.pdf.