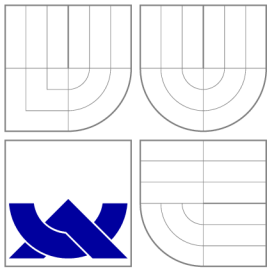
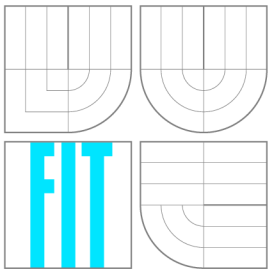


BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

AUGMENTED REALITY ON SMARTPHONES

ROZŠÍRENÁ REALITA NA SMARTFÓNOCH

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR
AUTOR PRÁCE
MAREK MINDA

SUPERVISOR
VEDOUCÍ PRÁCE
doc. Ing. MARTIN DRAHANSKÝ, Ph.D.

BRNO 2016

Brno University of Technology - Faculty of Information Technology

Department of Intelligent Systems | Academic year 2015/2016

Bachelor's Thesis Specification

For: **Minda Marek**
Branch of study: Information Technology
Title: **Augmented Reality on Smartphones**
Category: Computer Graphics

Instructions for project work:

1. Study the literature and overview technologies and approaches for augmented reality, photorealistic visualization and smartphone applications in this area.
2. Study and compare available framework packages for augmented reality on smartphones.
3. Design an application for rendering buildings and architectural objects into the augmented reality using a smartphone.
4. Implement the proposed application for Android OS using tools and framework packages mentioned in point 2.
5. Summarize the results and propose future directions of your work.

Basic references:

- Nagy, M.V.: *Utilizing Building Information Models with Mobile Augmented Reality and Location-Based Services*
- Kronander J., Banterle F., Gardner A., Miandji E., Unger J.: *Photorealistic rendering of mixed reality scenes*
- Rossi M.: *Methods and application for photorealistic rendering and lighting of ancient buildings*

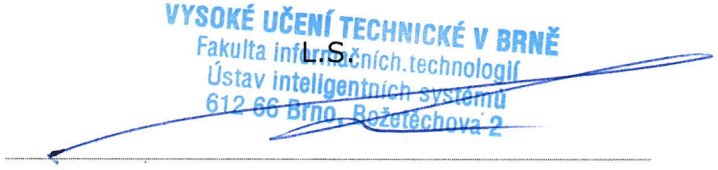
Requirements for the first semester:
Items 1 and 2 of the specification.

Detailed formal specifications can be found at http://www.fit.vutbr.cz/info/szz/

The Bachelor's Thesis must define its purpose, describe a current state of the art, introduce the theoretical and technical background relevant to the problems solved, and specify what parts have been used from earlier projects or have been taken over from other sources.

Each student will hand-in printed as well as electronic versions of the technical report, an electronic version of the complete program documentation, program source files, and a functional hardware prototype sample if desired. The information in electronic form will be stored on a standard non-rewritable medium (CD-R, DVD-R, etc.) in formats common at the FIT. In order to allow regular handling, the medium will be securely attached to the printed report.

Supervisor: **Drahanský Martin, doc. Ing., Dipl.-Ing., Ph.D.**, DITS FIT BUT
Beginning of work: November 1, 2015
Date of delivery: May 18, 2016



L.S.

Petr Hanáček
Associate Professor and Head of Department

Abstract

Augmented reality is a concept of supplementing a real world by a computer-generated content. Thanks to the development in capabilities of mobile devices, there are countless opportunities to use this concept on smart-phones. This thesis researches how current Android devices perform in 3D photo-realistic rendering and its usability for Augmented reality applications. Based on comparison of available frameworks, Wikitude SDK has been chosen to implement application for rendering buildings into AR. Geolocation is used for tracking purposes.

Abstrakt

Rozšírená realita je koncept dopĺňania reálneho sveta o počítačom generovaný obsah. Vďaka pokroku v oblasti mobilných zariadení je mnoho možností využitia tohto konceptu na Smartfónoch. Táto práca skúma schopnosť súčasných zariadení s operačným systémom Android zobrazovať fotorealistický obsah a jej využitie v aplikáciach s rozšírenou realitou. Na základe porovnania dostupných frameworkov, Wikitude SDK bol zvolený pre implementáciu zobrazovania budov do rozšírenej reality. Na účely trackingu je použitá geolokácia.

Keywords

Augmented reality, Smart-phone, Rendering, Photo-realistic Rendering, Android operating system, Tracking, Rendering of Buildings

Klíčová slova

Rozšírená realita, Smartfón, Zobrazovanie, Fotorealistické zobrazovanie, Operačný systém Android, Tracking, Zobrazovanie budov

Reference

MINDA, Marek. *Augmented Reality on Smartphones.* Brno, 2016. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Drahanský Martin.

Augmented Reality on Smartphones

Declaration

Hereby I declare that this bachelor's thesis was prepared as an original author's work under the supervision of doc. Ing. Martin Drahanský, Ph.D. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Marek Minda
July 26, 2016

Acknowledgements

I would first like to thank my thesis supervisor Assoc. Prof. Martin Drahanský at Brno University of Technology for his consultations and advices in both organisational and technical field. I would also thank Dr. Miglena Dontschewa from Fachhochschule Vorarlberg for help with choosing the topic of this thesis, along with Patrik Jost, BSc MA, from Fachhochschule Vorarlberg for his valuable advices about implementation of application. Lastly i would like to thank my friend, Martin Majna, student of Architecture at STU Bratislava for providing 3D models of buildings for testing purposes.

Contents

Chapter 1

Introduction

Augmented reality (AR) is a concept that was introduced in the 60s of 20^{th} century, but since then, no major applications of AR emerged. It was mostly because of the Hardware and also Software demands of AR setup, but the situation has changed in recent past, with success of smart-phones. Modern smart-phones are equipped with all necessary hardware and also enough computational performance to run software for Augmented Reality.

Following chapter defines augmented reality and all the requisites needed to accomplish it. In the 3^{rd} chapter, available software packages and frameworks for augmented reality are discussed and compared. 4^{th} chapter describes different approaches for Tracking - composing real world image with the virtual one. Following chapters describe application design, implementation and testing.

This thesis overviews modern approaches in Augmented Reality on smart-phones and rendering capabilities in this field. The goal is to render a 3D model of building or other architectural object on smart-phone and visualize it into a real-world image captured by smart-phone's camera using one of the AR frameworks.

Chapter 2

Augmented reality

Augmented Reality (AR) [4] is a technique where users are allowed to see the real world, with virtual objects superimposed upon or composed with the real world.

Unlike Virtual Reality (VR) [4], where the whole image is rendered, AR gives additional information as an overlay to the vision of a real world. Possibilities of use are limitless. From the simplest text information (e.g. current time, temperature or speed), to the complex photo-realistic imagery.

Creation of an AR system is a challenging task in means both of hardware and software, but the situation improves rapidly with modern technologies, especially in the mobile devices field.

2.1 Definition of Augmented reality for this work

As definition of AR suggests, the virtual content should be overlapped onto real-world scene. Goal of this work is to create an AR application for smart-phone, which is a device that does not allow creating an augmented reality application corresponding to it's definition. Technically, image taken by the smart-phone's camera is also rendered, making it the virtual reality. However, the camera image is realistic enough to be considered as a real scene, which allows applications using camera to be considered as Augmented reality.

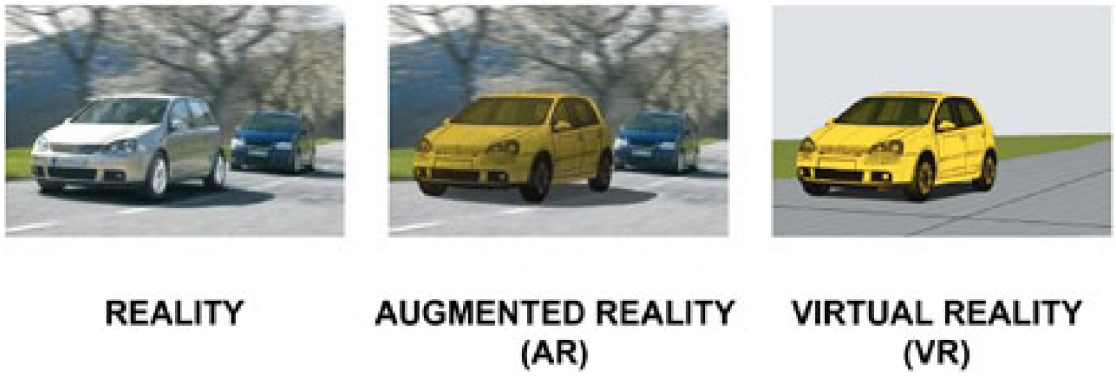


Figure 2.1: Difference among Reality, AR and VR. [15]

2.2 History of Augmented Reality

In 1968 Ivan Sutherland [19] constructed first headset, which rendered different images to both eyes, stereoscopically, in result of user perception as a 3D object. This headset is showed in fig. 2.2. The headset was primitive both in terms of rendering quality and the user interface. As the augmented object, primitive wireframe rooms were rendered.

To achieve credible user experience of AR, all rendered objects should display dependent on the head position, rendering them steadily into the real world.

Paul Milgram and Fumio Kishino introduced the „virtuality continuum“, which is an axis from a real world to a Virtual Reality. AR is placed on this axis to better represent its purpose. Fig. 2.2.

Since then, no major breakthrough was accomplished in this field, due to limitations of the computing power in the 1970s and 1980s. Situation has changed with the emerge of the smart mobile devices, that bring necessary computational power and desired hardware to realize the AR rendering.

The term Augmented Reality was introduced by Thomas Caudell and David Mizell in 1990 [19].

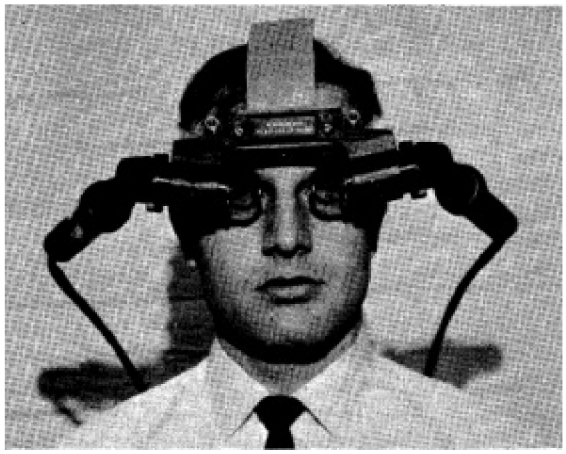


Figure 2.2: Head-mounted system with a display in front of each eye. [19]

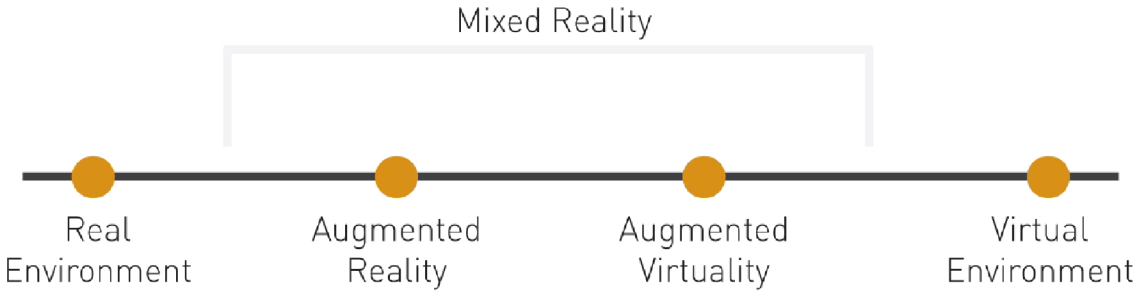


Figure 2.3: Simple version of the virtuality continuum. [4]

2.3 Goals of Augmented reality

Virtual and Augmented reality should persuade all of human senses [1] to get the best and indistinguishable experience for users. Photo-realistic rendering for sight, high-quality audio surround sound for hearing, Aroma generators for smell and taste. The most difficult task is to simulate ability of manipulating with objects by touch, so the user can grab augmented object in hands, feel it and put it back. These requirements are more important for the VR content, but are also needed in AR to give trustworthy experience. This thesis focuses on rendering buildings into the Augmented reality on smart-phone's screen, so the visual, rendering, part is the most important one.

2.4 Software requirements

An ideal AR system should integrate the rendered image into real world unrecognizably for user, but that is, for now, impossible to achieve, so the systems can be compared and ranked based on completing this goal. All input data from different sources must be integrated into one coordinate system. This process is called Image registration [19]. When rendering 2D objects into AR, you just render it relative to the environment. This is mostly used for text and pictures. When placing 3D objects, you have to attach them absolute to the environment, so they don't turn as you move the camera around them. To ensure a stable position of a rendered object, process called Tracking is used. There are two options to achieve it:

- Bounding to an anchor real life object
- Calculation from the position and rotation of the camera

Using the first option, the object is placed based on patterns in the image, e.g. a QR code sticker. In this case, image-recognition must be implemented to detect patterns in the input image.

When placing the object with the second method, you have to render it into the real world coordinates. This case is more challenging, as you have to deal with accuracy and reliability of the sensors.

When the augmented object is successfully placed into the system, we need to ensure it's realistic visualisation. Real-Time Rendering algorithms are used for this purpose. Lighting, shading and object interference with real scene, e.g. dropping shadows, are required to achieve photo-realistic quality of the augmented object. Complete AR image is then finally rendered to user.

2.5 Hardware requirements

Depending on the specific application, various sensors and peripherals are required to achieve the AR. For example, device that gives you information about actual time, needs only the transparent head-mounted display and clock, but rendering buildings into real world is a different story. You need precise location of the camera (location services), rotation of the camera (accelerometer + compass), camera itself, and a display to show final rendered mixed image of the AR. Resolution of the device's display should also provide sufficient pixel density and refresh rate. To fulfill this requirements, used device must have

enough computational power. Quality and performance of the device's camera is also important to meet the definition of AR as much as possible. Thanks to the recent progress, all of mentioned hardware is mounted on a common smart-phone in your pocket.

2.6 Mobile devices

After years of development in the mobile computing and mobile devices field, hardware equipment and computing power seems to be sufficient to run AR applications on these devices.

First of the devices used to implement this thesis's application on is LG G3. LG G3 (model name LG D855) is a LG's flagship model launched at 28^{th} of May, 2014. This device was the first smart-phone with qHD resolution (2560×1440) display with 538 pixels per inch, hitting on the Retina [14] value, providing sharp image quality. On the other hand, resolution that high, unfortunately kills the graphic performance in 2D and also 3D applications. Camera of this device is equipped with laser sensor, granting faster focusing on objects. Integrated A-GPS Antenna along with Glonass gains device's location lock immediately in the clear sky-view locations, with 4.5-6m accuracy.

The second device used is NVIDIA SHIELD Tablet in its 32GB LTE version, launched on July 29^{th}, 2014. Nvidia intended this device to be a terminal for streaming and decoding high-quality and low-latency video content, which is performance demanding task. To achieve this, SHIELD Tablet is equipped by GK20A (Tegra K1) GPU, which is true-desktop GPU with only one SMX block [7]. Display ppi is lower than the LG G3, same like the Camera resolution and quality. To particular values and specs, see Table 2.1. Accuracy of the location on this device is more accurate than LG G3's, providing 3.5-4.5m.

According to specifications of these devices, it's possible to expect some results of the application. LG G3 should provide more sharp rendered image, overlapped on more detailed camera picture, with a risk of not sufficient frame-rate due to lower performance. Location based tracking is expected to be also less accurate. NVIDIA Shield Tablet with less dense display, would not be able to provide image as sharp as the LG G3, also with less detailed camera, but the computing performance is at another level, also with position lock accuracy.

Accuracy of the location on both devices was measured by GPS Info [26] application.

device	LG G3	Nvidia SHIELD
Screen size	5.5 inches	8 inches
Resolution	2560×1440	1920×1200
Pixel density	538 ppi	283 ppi
Display technology	IPS LCD	IPS LCD
Camera	13 Mpix	5 Mpix
Processor model	Krait 400	ARM Cortex-A15
Processor cores	4	4
Processor frequency	2,500 MHz	2,200 MHz
Graphic processor	Adreno 330	GK20A
Positioning	GPS, A-GPS, Glonass	GPS, A-GPS, Glonass
Sensors	Acc, Gyro, Compass	Acc, Gyro, Compass
Android version	Android 6.0	Android 5.1
Launched	Q2 2014	Q3 2014

Table 2.1: Used mobile devices [22]

Chapter 3

Frameworks for mobile devices

Since the origin of the AR, many software development kits emerged. The older ones were implemented with an option of cloud computing to accelerate demanding calculations (e.g. CloudRidAR [32]). The biggest boom of AR frameworks occurred with increased performance of mobile devices. Current graphic accelerators in smart-phones are powerful enough to deal with real-time rendering of AR. The most advanced frameworks for mobile devices are Wikitude and Layar. Since the number of augmented reality frameworks is countless, it was impossible to try every one of them to compare and choose the best one as a part of this thesis. For this purpose, Augmented Reality SDK Comparison [3] was used. This comparison gives tabular comparison of more than 30 existing frameworks for Augmented reality. In the next section, requirements for the framework are described, and frameworks that meet those criteria are filtered.

3.1 Requirements

AR framework chosen for this thesis should meet these requirements:

- 3D object loader
- 3D object rendering
- Geo based rendering
- Device sensors
- Marker recognition
- Android OS

Implementation of the rendering algorithm is not a part of this thesis, because of complexity of such algorithm, requiring an SDK with built in 3D rendering engine used for visualising models. For that purpose, object loader is also necessary requirement for SDK functionality, also with support of commonly used 3D file formats that are able to store 3D object data. One of the tracking method must be used to compose rendered image over device's camera correctly. For that, data from geo sensors, accelerometer and digital compass are required. Last and very important condition is support for Android operating system.

When all of above mentioned requirements are combined, portfolio of suitable SDKs narrows significantly. Most of the available SDKs are without 3D rendering support. In result, there were only two SDKs that met given requirements: Wikitude [6] and Layar [11]. MetaIO [17] is also very robust SDK, but after their acquisition by Apple Inc., this SDK is for iOS only. In the next section, there is detailed description of Wikitude and Layar SDKs.

3.2 Wikitude

Wikitude GmbH is the world's leading mobile augmented reality (AR) technology provider for smart-phones, tablets and digital eye-wear. Company is based in Salzburg, Austria, and was founded in 2008. [6] Formerly this framework was aimed at AR browser [31], application that used information from maps and rendered it into AR using built-in camera app. Today, AR Browser integrates many other services using location-based information, such as Trip Advisor, Wikipedia, Restaurants and others, see 3.2. Since August 2012, Wikitude also implements algorithms for image recognition to use patterns as a tracking for rendering of Augmented reality objects. The newest feature of Wikitude SDK is 3D tracking, capable of scanning the 3D parameters of recorded scene, only by rotating the camera around an object.

Wikitude is a robust SDK and meets all of the given requirements for this thesis. Wikitude contains 3D object loader and renderer, however none of the common 3D file formats are supported. The Wikitude 3D Format (.wt3) file format is used instead. This in-house binary format uses compression to describe 3D content, to optimize loading times on mobile devices. In order to convert more conventional file formats into .wt3, Wikitude offers a desktop application called „Wikitude 3D encoder“ [30]. This application lets you import .fbx file format and then export it into a .wt3 file. You can also revise the loaded 3D model using built-in 3D viewer, to distinguish any anomalies before exporting to .wt3 file, as shown in 3.3.

Wikitude SDK uses geo sensors such as GPS and Glonass to determine device's location. The direction in which is the device facing is calculated by compass and accelerometer. Android and iOS operating systems are supported by Wikitude SDK.

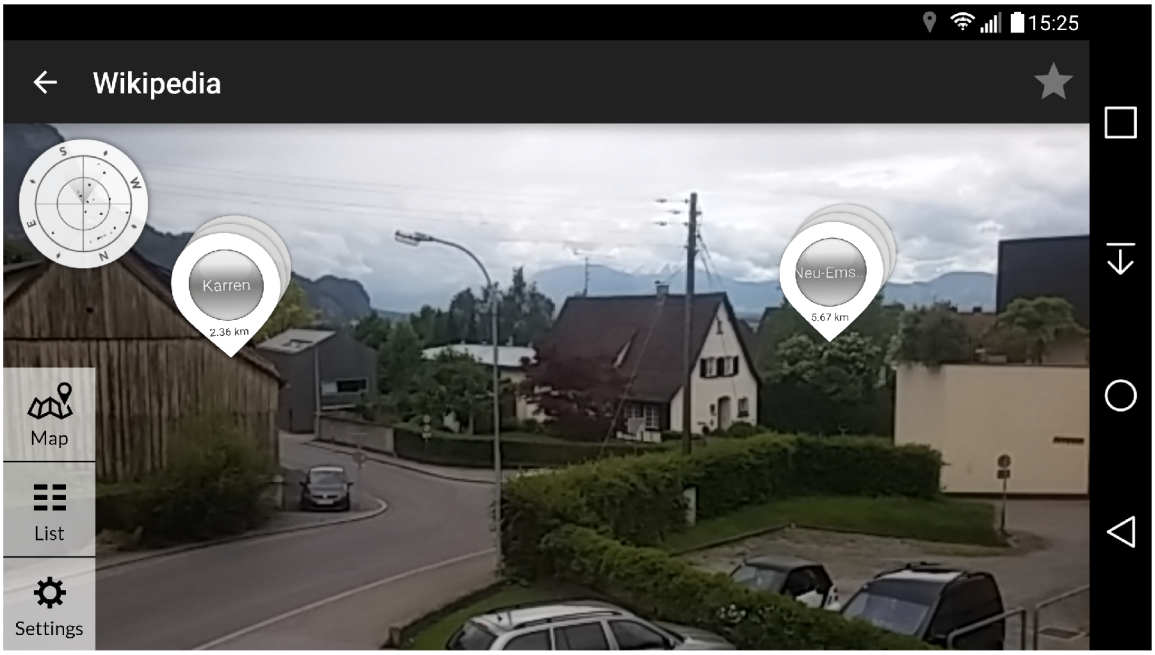


Figure 3.1: Wikitude browser screenshot.

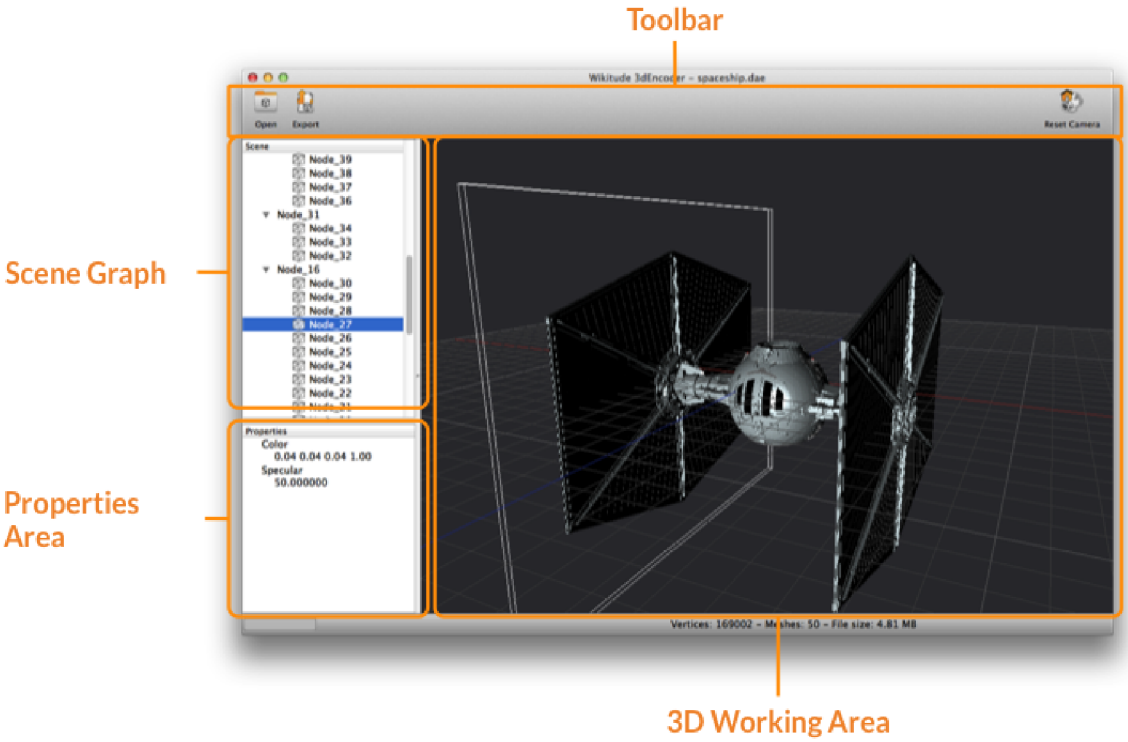


Figure 3.2: Wikitude encoder screenshot.

3.3 Layar

Layar was founded in 2009 as one of the first mobile augmented reality browsers for mobile devices that hit the market [11]. This company is based in Amsterdam, Netherlands. Similar to Wikitude, Layar also offers free application [13] to browse various data into AR on your smart phone or tablet, see 3.5. After creating a developer account on Layar website, you can access all the features that Layar SDK offers. This SDK is divided into small modules, called Layers. Each Layer implements one piece of functionality, i.e. 3D rendering, location services, image recognition and actions.

Layar SDK also fulfills required criteria for this work. Similar to Wikitude, Layar's object loader uses its own 3D file format called „Layar3D" with .l3d extension to store data describing 3D objects. Layar3D Model Converter [12] is a desktop application enabling import of .obj or .mtl 3D data files. After the import you can inspect the model ind 3D viewer, and view detailed information such as number of vertices, faces and materials, see 3.6. You can also use „Placement" tool to align your model to the map, for later showing it on desired location in AR application. This tool is important and could be useful for this Thesis. Afterwards, you can export 3D object into .mtl file.

To render building at the selected location, Layar application uses location services and device's accelerometer to compose the image accurately into the AR. For image tracking purposes, „Layar Vision" Layer is used. This tool uses detection, tracking and computer vision techniques to augment objects into the real-world scene. Layar SDK supports both Android OS and iOS, satisfying the operating system requirement.

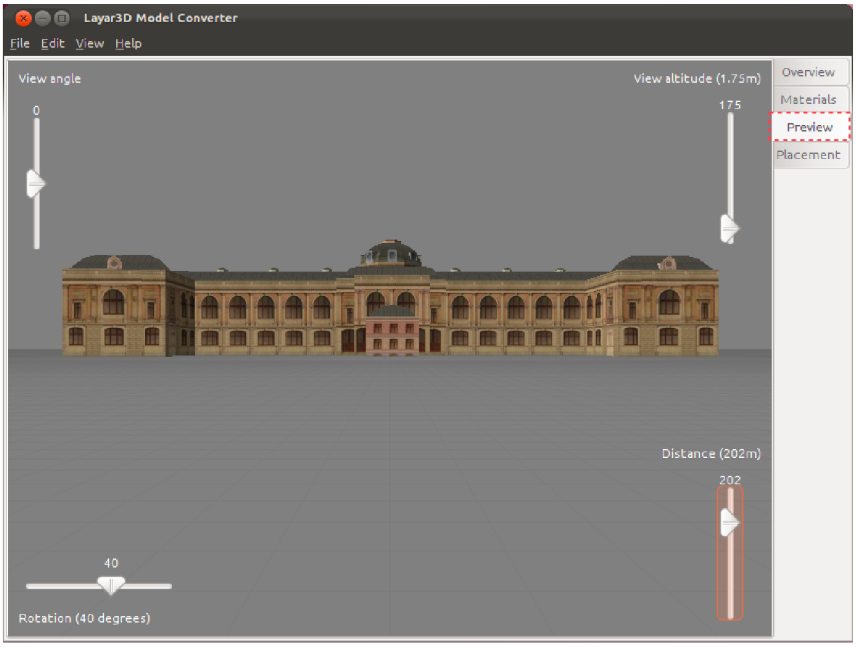


Figure 3.3: Layar 3D Model Converter screenshot.

3.4 3D File formats

3D data are used in many fields, with different purposes [16]. For example in medicine, mechanical-engineering, video animation, architecture etc. File format used to save data of

Figure 3.4: Layar browser screenshot.

a 3D scan of teeth needs only information about the shape and the volume of them, where character model used in video game needs to save information about shape, components, materials, textures and optionally animations of the character. Because of this diversity, many formats has emerged. This thesis application is aimed at architectural field users, so most of the loaded objects will be buildings. That means, following data needs to be stored in the file.

- Object geometry
- Materials
- Textures
- Animations (optionally in rare cases)

Another requirement for the file format is it's ability to be loaded by Framework used for the application. As both of the mentioned frameworks use in-house file formats, their converters needs to be used. Wikitude's encoder can import only .fbx format, [30] while Layar's converter can load Wavefront (.obj or .mtl) files [12]. Users that want to load their model into this application and are unable to export it in one of the mentioned formats, are bound to use some of the 3^{rd} party file format converters.

FBX (Filmbox) is a proprietary 3D file format, that was accuired by Autodesk Inc in 2006. Content is normally saved in binary format, but saving in ASCII is also possible. This format can store information about cameras, lights, meshes, NURBS and other elements of the scene [8]. All of the Autodesk applications, that works with 3D, are capable of exporting their 3D data and scenes into the .fbx file format.

OBJ is an open geometry definition file format developed by Wavefront Technologies, adopted by other 3D application vendors. MTL is than used as a plug in for OBJ files, describing used materials [29]. Since this format is open, many 3D applications support exporting to the .obj file format.

Both of the mentioned 3D file formats are used across all of the common 3D programs (AutoCAD, ArchiCAD, Revit) so there should not be a problem for users to load their model into this thesis's application. In case of no option of exporting the model by particular program, 3^{rd} party software should be used for conversion.

3.5 Comparison and results

Both Wikitude and Layar were candidate frameworks to be used for implementation of this thesis application. Both of them are web based frameworks implemented in Javascript language. Both frameworks use their owm 3D file formats, so user is bound to use software to convert to these files. Layar is, out of the box, more suitable for geographical use for architecture, thanks to the built-in location picker. On the other hand, Layar framework is not that good configurable as Wikitude framework is, resulting in more easy to use solution, but Wikitude offers more customization to their framework. Wikitude framework also offers more options to interact with objects in Augmented space and also have a lot more content in their support forums. Considering all the mentioned pros and cons of both framework packages, Wikitude has been chosen to implement this thesis application.

Chapter 4

Tracking

To achieve trustworthy Augmented reality experience, the rendered augmented object must be overlapped properly on the camera image. To achieve the Image registration, process called Tracking [5] is used. Tracking can be achieved by multiple ways, differing in input data used for it. The ideal tracking algorithm should keep the augmented object overlapped in the way that is not recognized as augmented. Rendered image should stick exactly at desired place, even when the observer walks around to see it from different angles. Tracking approaches can be divided into two groups, depending on usage of input image from the real scene. Non-visual tracking is able to work without processing the image of real world, while Visual tracking is dependent on recognizing patterns in the input image to overlap the augmented object. In the following section, both approaches are described in detail.

4.1 Non-visual Tracking

Tracking without the visual input must use other data to realize the Image registration. Real world, camera and augmented object have to be projected into one common coordinate system. Knowing the location and rotation of camera is essential for determining angle of view of the augmented object. Desired position of the augmented object is also important.

Location of the camera, in terms of this thesis, location of the smart-phone is measured by many sensors. It is possible to get the location by GSM network, WiFi networks or by global positioning systems provided by satellites from space. The last mentioned method is the most accurate one. Setting up a location of the augmented object with Non-visual tracking is done by setting object's desired location by geological coordinates of the real-world. So the one common coordinate system for Image Registration is Geological longitude, altitude and latitude.

When the object's desired position is set, we need to measure the location and rotation of the camera, in this case, the mobile device. For the device's location measurement, built-in Geo-location chips are used, usually GPS, GLONASS and GALILEO [25]. All of these location-measurement systems work on the same principle. Satellites in space, orbiting earth, are transmitting signals with time information, enabling the destination device's antenna to receive this signal and compute distance from the satellite. To gain 3D location lock, signal from at least 3 satellites is needed. Location lock accuracy increases with number of satellites, example shown in Fig 4.1 . Because of the distance that the signal travels, accuracy of this method is usually less than 3 meters. With worse weather conditions or determination in areas with narrow visibility of sky, accuracy gets way worse. How-

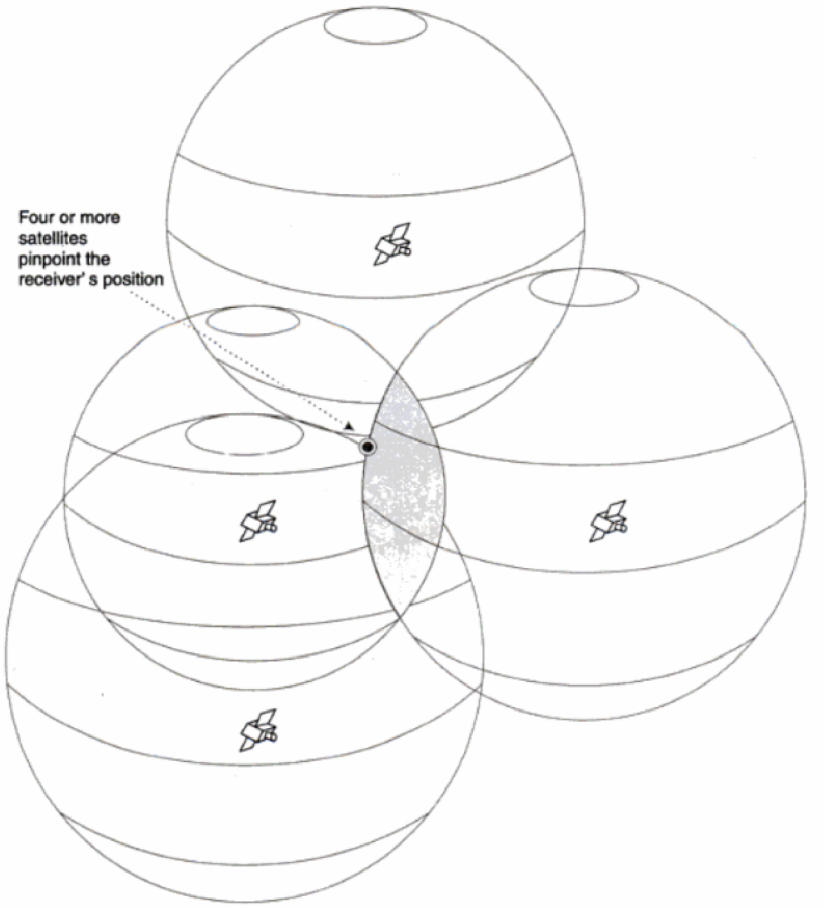


Figure 4.1: 3D position determination using 4 satellites [25].

ever, combining multiple navigating systems into one location lock, improves the accuracy. Smart-phones are also equipped with A-GPS Antennas (Assisted GPS), when Internet connection is used to download latest reported location of the satellites to get more accurate results and faster location lock.

When the device's location is determined, we need to measure skew of the camera, to do the Image registration properly. For this purpose, trio of sensors Accelerometer, Gyroscope and Digital Compass are used. Accelerometer chip measures acceleration of the device in 3 axes, providing information about slight movements of the device. Gyroscope calculates a tilt of the device, also on 3 axes, and Digital compass determines relative rotation of the device from the Northern magnetic pole of Earth.

4.2 Visual Tracking

Visual tracking methods do not use information from other sources. Only data input needed for achieving the Image registration is the camera image itself. Algorithms for this type of Tracking must implement image recognition. For this purpose, Markers are used [10]. Markers are patterns, usually objects placed in the real-world scene, that are used to determine position of the augmented object. Most common case is usage of QR-codes, becasue they are easily recognised in the image, see Fig. 4.2. For more accurate placement,

Figure 4.2: Augmented object position determination using QR-code Marker.[18]

using of 3D Markers is also possible.

4.3 Markerless Visual Tracking

Markerless Visual Tracking is a combination of Visual and Non-Visual Tracking, when the input image is used without Markers,along with data from device's sensors to achieve the Image registration. In this case, Smartphone camera's rotation is used as a simulation of Stereoscopic vision to get the depth map of the scene, enabling recognition of 3D objects in the real-world scene [2]. Wikitude's 3D tracking technology uses this approach to achieve Markerless Tracking.

Chapter 5

Real-Time Rendering

Rendering is a process of transforming 3D models and scenes saved in the computer into 2D image that is subsequently showed on a displaying device, usually the monitor [27]. Rendering is a most complex and performance demanding segment of the Computer Graphics, used in many disciplines (Medicine, Architecture, Mechanical Engineering, Video Editing etc.). Each of this disciplines demands different parameters from the rendered image (Image quality, Rendering Speed) resulting in many rendering techniques emerged. Real-time rendering is rendering that is able to interact with user in a way, that user does not recognize individual images delivered to him, in hand with a minimal latency. This requires fast passage through the Rendering Pipeline, which is very performance-demanding task, exceeding the CPU performance capabilities. This need was satisfied by using a dedicated hardware unit used for geometric computations, Graphics Processing Unit (GPU).

5.1 Rendering Pipeline

Pipeline concept is used in many different forms, where the process is dividable into separate stages to prevent lag of the whole system [27]. Rendering Pipeline, also called Graphics Pipeline or Graphics Rendering Pipeline is divided into 3 conceptual stages to increase the speed of entire rendering : application, geometry and rasterizer, see Fig. 5. Each of the stage can be divided into more stages, meaning that every stage is also a pipeline itself. Some parts of the pipeline can be parallelized in order to meet high performance requirements. Speed of a rendering pipeline is measured in fps (frames per second) and is determined by the slowest stage. This speed usually varies, according to complexity of the computed scene, which is changing frame to frame, as user interacts with the application.

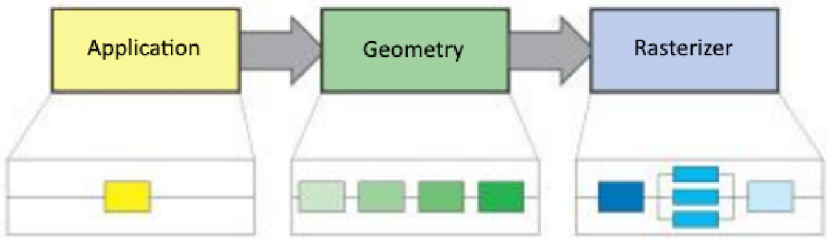


Figure 5.1: Rendering pipeline stages.[27]

Application stage of the pipeline is driven by CPU software, enabling parallel processing thanks to the multiple threads of CPU. Various tasks are run in this stage: collision detection, global acceleration algorithms, animation, physics simulation (not necessarily), and many others, depending on the type of application. Geometry stage computes transforms, projections, etc. Determination of what how and where should be drawn is performed, usually on a Graphics Processing Unit (GPU). The last, rasterizer stage is processed completely on GPU, rendering the data outputting from the previous stage, along with any per-pixel computations desired.

Usage of the dedicated hardware for computing graphics is necessary to achieve fluent real-time rendering without lag, which is fundamental also for Augmented Reality applications. Thankfully, all modern smart-phones and tablets are equipped with powerful GPUs, enabling Augmented Reailty experience on mobile devices.

5.2 Photo-realistic rendering

To achieve believable Augmented Reality experience, the augmented object should be rendered in the best available quality and realness. For this purpose, Photo-realistic rendering techniques are used. Photo-realistic Rendering, also called Unbiased Rendering is a technique that renders a scene physically precisely, usually by tracing a light using per-pixel algorithms. Unbiased rendering methods include:

- Path Tracing
- Light Tracing
- Bidirectional Path Tracing
- Metropolis light transport

One of the most widely used techniques for Photo-Realistic rendering is Ray tracing [21]. Ray tracing is a per-pixel algorithm, that uses Path Tracing technique, casting rays from camera into space, to determine pixel colour. It is also used in z-buffer implementations of renderers to achieve collision detection of objects and mouse click selecting them. Mouse click is bound to one pixel, from which a ray is casted into the space, picking the first object it hits.

In rendering, Ray tracing is straightforward in computing shadows and reflections. Basic Ray tracing algorithm is following:

```
for each pixel do
    compute viewing ray
    if (ray hits an object with t  ∈ [0,∞) ) then
        Compute lighting equation and set pixel to that color
    else
        Set pixel color to background color
```

This example computes color of the pixel only according to the material and applying non-tracing lightning model (i.e. Phong model [28]) to the pixel of the object's surface. In result, image quality is comparable to biased renderers. Full Ray tracing engine computes the lighting of a scene by rays. After the ray hits an object, it reflects and tracing the ray iteratively until the ray hits a light source. Same principle is used for computing shadows.

When the ray, coming from point on surface of the object hits another object on its way to the light source, computed point is in shadow. This technique is physically accurate, because it describes and compute behaviour of light in a real scenes. Result images are in special conditions, hard to tell apart from photos. Nvidia Iray [20] is a photo-realistic rendering engine that generates photorealistic imagery by simulating the physical behavior of light and materials, see Fig 5.2.

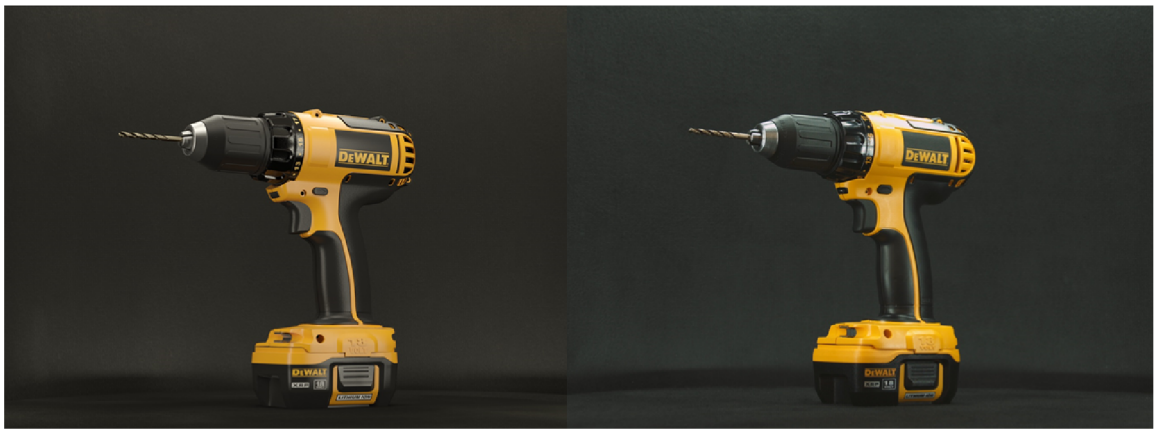


Figure 5.2: Image rendered by Nvidia Iray, the left one is rendered [20].

5.3 Performance demands

Nvidia Iray rendering engine, mentioned in previous chapter, is able to run only on desktop computers with powerful graphics card, and still not in real-time.Ray tracing is a per-pixel algorithm, so its computational demands increase rapidly with higher resolutions. On mobile devices is so far not possible to achieve this quality of rendered image. Real-time rendering is also necessary for Augmented reality application, and also reserve of computational power for processing Camera Image and device's sensors.

To determine how 2.6 used for this work perform in Ray tracing rendering, benchmark Android application GP2Mark [24] was used. This application implements GPU Ray tracer, rendering scene with spheres moving on top of the chessboard plane. LG G3 with it;s qHD screen was not even able to run the application natively. After lowering the resolution to fullHD, it was able to render the scene at 12 frames per second (fps). This is far from being usable for Augmented reality application. On the other hand, Nvidia SHIELD tablet with it's desktop based graphics card outperformed LG G3, giving stable 60fps at it's 1920×1104 resolution.

Performance results from the Ray tracing benchmark signifies that so far there is no chance for common devices to run real-time rendering of Photo-realistic scenes aimed to Augmented Reality, but their performance is converging to satisfy the demands.

5.4 Vulkan API

Hardware is not only field that is going through a significant progress. In hand with hardware, software APIs and drivers of graphical processing units undergone a lot of improvement as well. One of the most important change is the Vulkan API. Vulkan is a

low-level API for graphics and compute hardware. It is a set of commands that allow the specification of shaders, kernels and data used by them [9]. From another point of view, Vulkan is a Rendering pipeline that has some programmable stages and some state-driven fixed-function stages that are invoked by specific drawing operations. This customizations enable a lot more control of the graphics computation hardware, allowing them to write more optimized code.

Using the Vulkan API reduces the number of draw calls sent to the CPU, resulting in significant reduce of the CPU overhead. This solution provides performance boost on every single platform in existence, even on mobile devices. Android devices are also supported, but only ones with hardware support of OpenGL ES 3.1 and newer versions. From mobile devices used for testing this application, only NVIDIA Shield Tablet meet this requirement.

Vulkan API is still relatively new, so there are only few applications implemented and the situation is even worse on mobile devices. Benchmark tests, however, are showing a huge potential of this API and we will definitely see a lot more applications using this API in a near future. Based on recent benchmarks, performance boost of the Vulkan API can reach more than 80% over OpenGL solutions [23].

Chapter 6

Application design

Application for this thesis was programmed natively for Android operating system, however the chosen SDK - Wikitude is implemented as a Web application launched from Android application environment, providing easy implementation also on another operating systems. Android package name of the application is „cz.vutbr.xminda00.bpmindaar“ and name of the application is „AR Buildings viewer“. After the first launch, user is asked to allow following application permissions, that are necessary for proper runtime:

- Internet access
- Device location
- Camera
- Accelerometer
- Compass

Internet access is necessary for communication with the web server, to store 3D models and their desired locations, along with Google Places API used for the Place Picker. Elevation of the picked coordinates is also evaluated online using Google Play services.

Device's location represents location of a camera in the AR space. In this application, device's location is calculated by GSM, Wi-Fi and geo-sensors (GPS,Glonass) combined in effort to provide most accurate lock possible. When the application cannot get proper location lock (using only GSM and Wi-Fi without geo-sensors), user is informed that the low-accuracy mode is used, meaning the elevation of the camera cannot be calculated, resulting in elevation of the model being evaluated absolutely in the same altitude as the device is.

Without device's camera, realising of the Augmented reality is impossible. Rendered image is overlapped to the image of camera. In this application, back camera is used, because of the better quality on vast majority of mobile devices, in hand with much bigger convenience while using the application.

Accelerometer and Compass are required to compute rotation of the camera, which is also necessary to provide trustworthy Augmented reality experience.

Using all of these sensors and device's computational performance at once, almost during the whole application runtime, results in huge power consumption and low battery life. This is caused mostly by the full brightness of the screen necessary in outdoor environment, in hand with location locking sensors, and almost 100% CPU usage for the rendering.

This application works on every Android device with OS version not older than Android 4.3 and on every screen size except Android wear devices. Both landscape and portrait mode are enabled to use for this application.

After the application launch, main menu with following options appear:

- AR View
- Pick a Location
- Select a Model

Main menu is a simple list with mentioned options. By taping any of the items, desired sub-menus with different functionalities are opened. In the following sections, all of these sub-menus are described in detail.

6.1 AR View

AR View is a full-screen representation of this thesis application main functionality. Wikitude SDK is used to render user defined model at an user defined location over the image of the device's camera. Device's location and sensors determine if the model is in the field of view of the camera. If not, an arrow indicating position of the model is displayed on the corresponding border of the screen. At the first launch of the application, when the result is not displayed properly, digital compass calibration should be done, by moving the device in the shape of number 8. Building is then placed in the desired location, but in most cases it's not rotated in a way that the user demands. For this purpose, rotation of the model is implemented and triggered by touching the building model. Touch the model again to stop the rotation at the desired moment. AR View is shown in , see Fig. 6.1:

Figure 6.1: Screenshot of the Augmented reality view of a building.

6.2 Pick a Location

After this item from the main menu is selected, User interface for picking a location is displayed. This Google Places API allows user to pin-point an exact location nearby, based on device's actual location. It also allows to search for a place or address by name, using the search bar. After the location is picked and confirmed, geological coordinates (latitude and longitude) of this place are set as a placement of the building for AR View. User is informed about success of this operation and redirected back to the main menu. Third necessary parameter for proper rendering, elevation of the desired place, is determined using Google elevation API and stored with the coordinates for the further use. Location picker is shown in Fig. 6.2

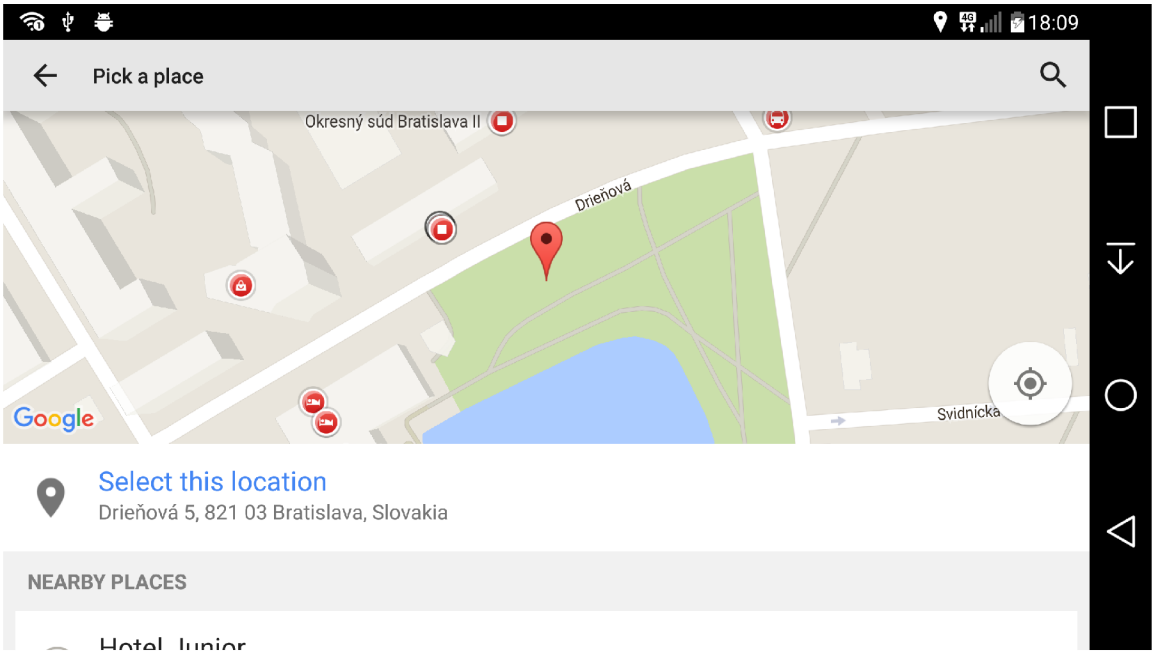


Figure 6.2: Screenshot of the location picker.

6.3 Select a Model

In this sub-menu, user can choose the desired 3D model to be rendered in AR View. These 3D models are located on the web server. Models are stored in Wikitude's `.wt3` file format to optimize loading times. User is able to use Wikitude 3D encoder software to load `.fbx` file format and then export it to the `.wt3`. To ensure precision, location selected by Google place picker, is the $(0, 0, 0)$ point in the coordinate system of the 3D model.

Chapter 7

Application implementation

Application was developed using Android Studio 2.1.2 with `minSdkVersion 16` and `targetSdkVersion 23`, meaning this application is targeted to operating system Android 6.0 and minimal required version is Android 4.3. Java development kit is necessary to build Android applications, in this case, $jdk1.8.0 - 91$ was used. For Augmented Reality implementation, Wikitude SDK version 5.1.4 was used. Android application was programmed in Java language, with XML descriptors, Wikitude Framework functionality part was coded in HTML and JavaScript, and the simple Webserver is implemented in PHP language. In following sections, all the important parts of the implementation are mentioned.

7.1 Android application

Android application implementation is located in `/app` folder. Application is built using Gradle Scripts, where all the used external APIs and SDKs are imported.In file `app/src/main/AndroidManifest.xml`, application name and package name are set, along with SDK version requirements and application permissions. Further, all application Activities are mentioned, and their parameters (fullscreen, Arview...) are set. in meta-data tag, application key for Google Places API is defined.

Folder `app/src/main/java/cz/vutbr/xminda00/bpmindaar/` contains all the `.java` source code files of classes implementing the functionality of the application.

ArActivity class

`ArActivity.java` file implements the AR View functionality handled by Wikitude's `architectView`. Following methods are implemented in this class:

- `public void onCreate( final Bundle savedInstanceState )`

  Wikitude's `architectView`, which is a view holding Wikitude templates for viewing AR content is configured in this method. Device's location data are sent to this view, provided by `LocationProvider` class.

- `protected void onPostCreate( final Bundle savedInstanceState )`

  After the succesfull initialization in `onCreate` method, wikitude's `architectView` template `arview/index.html` is launched in this method. This template is located in `assets` folder.

- `protected void onResume()`

  This method and the rest of the methods implemented in `ArActivity` class override Activity's lifecycle events and pass them to the `architectView` to be handled correctly.

- `protected void onPause()`

- `protected void onStop()`

- `protected void onDestroy()`

- `protected void onLowMemory()`

LocationProvider class

This class implements a simple solution of location determination, which uses `FINE_LOCATION` resulting in the heaviest power consumption, which is necessary for continuous real-time location lock of the device. This class implements communication with `ArActivity` and it's behaviour when the host-activity is paused or resumed. Following methods are implemented:

- `protected void onPause()`

  This method just pauses location updates when the focus is lost, due to power saving reasons.

- `protected void onResume()`

  Location determination permissions are checked here, if yes, last known location measured by GPS or Network/Wi-Fi positioning is sent to `ArActivity` class. If not, user is informed about restricted location access.

MainActivity class

Implementation of the Main Activity of application. This activity is shown after the application is launched. Here you can find an implementation of these methods:

- `protected void onCreate( Bundle savedInstanceState )`

  Firstly after the application is launched, user is asked to grant permissions to access device's `CAMERA` and `FINE_LOCATION`. Main menu's `ListView` is filled with items right after. Cache of the `architectView` is also wiped here using `deleteDirectoryContent` method.

- `protected void onListItemClick(ListView l,View v,int position,long id)`

  After clicking on an item in the Main menu, Activities are called from this method.

- `private static void deleteDirectoryContent(final String path)`

  This method deletes content of a directory given by path. This method is called in `OnCreate()` method.

PlacePickerActivity class

This class implements behaviour of the Place picker. After choosing this option in main menu, Fullscreen content with map and search bar shows up, focused to actual device's location. This user interface is implemented by Google Places API, that uses Google Play Services (imported in Gradle build scripts). Special API key was retrieved from Google to enable this functionality. Following methods are implemented in this class:

- `protected void onCreate(Bundle savedInstanceState)`

  This method is called when Activity is created. Place Picker builder is used to build the request, and then the Place Picker API is called.

- `private static class Params`

  This nested class is used to store geographical data (latitude and longitude) received from Place Picker, into one object to simplify further manipulation. It also contains method to set the values.

- `public class Post extends AsyncTask<Params, Void, String>`

  This class is used as a wrapper for `doInBackground` method, providing out of the main thread networking operation, which is given by Android operating system conventions.

- `protected String doInBackground(Params... params)`

  This method takes geographical data in `Params` object as an input, and then uses OkHttpClient to send these data via `HTTP POST` request to the web server. It is necessary to store this data outside the aplpication, as long as the Wikitude framework can not be loaded with launch parameters when running in an non-web application.

- `protected void onActivityResult(int requestCode, int resultCode, Intent data)`

  This method is called after leaving the Place Picker. If the location was successfully chosen, latitude and longitude are stored into `Params` object and then sent to the `doInBackground` method. After this, the Main Menu is opened.

WikitudeSDKConstants class

Wikitude's SDK key is stored in this class. It needs to be loaded dynamically to successfully obey the API key rules given by Wikitude.

SetModelActivity class

This class implements selection of a 3D model of building to be rendered in the AR view. User's selection is then stored on the web server for further use.

- `protected void onCreate(Bundle savedInstanceState)`

  This method initializes the content view for List layout showing available 3D models.

- `protected void onListItemClick(ListView l,View v,int position,long id )`

  After clicking one of the listed items, id of the 3D model is sent to the webserver. After that, user is informed about success of this action and returned to the main menu.

- `private static class Params`

  This class saves and implements setter for the id of the 3D model. Instance of this class is used as a parameter to be sent via POST request.

- `public class Post extends AsyncTask<Params, Void, String>`

  This class is used as a wrapper for `doInBackground` method, providing out of the main thread networking operation, which is given by Android operating system conventions.

- `protected String doInBackground(Params... params)`

  This method implements sending of data via HTML POST, using OkHTTP API.

7.2 Wikitude Framework architectView

Instance of Wikitude SDK for displaying augmented reality content. ArchitectView is an overdriven Android view displaying template stored in `app/assets/arview/index.html`. As Wikitude is a web framework, this template is written in HTML with JavaScript functionality in `js/3dmodelatgeolocation.js` and displayed as a webpage.

- `createModelAtLocation: function createModelAtLocationFn()`

  This function is called on initialization. Files with the stored geolocation and 3Dmodel in Wikitude's `.wt3` file format are loaded from the webserver. Animation for rotation of the building in AR View is configured, and the indicator arrow image is loaded here.

- `worldLoaded: function worldLoadedFn()`

  This function removes loading bar from the user interface after all the necessary data is successfully downloaded.

- `toggleAnimateModel: function toggleAnimateModelFn()`

  This function handles toggling of the animation of rotation of the building's 3D model rendered in the Augmented reality view on touch.

7.3 Web Server

Web server imlpemented in PHP scripting language handles POST requests from the Android application. Data about geographical location and 3D model are stored on the server for further providing to the AR View of the application. This information is stored in `location.txt` file. First line is geographical latitude, second longitude. Third line contains elevation in meters of place given by coordinates. This value is computed using Google elevation API at `http://maps.googleapis.com/maps/api/elevation/`. Last line of this file contains 3D model id to be rendered in the AR view.

Chapter 8

Application testing

In this section you can find how the implementation of the application performs and meet the required criteria to achieve Augmented reality experience, mostly image quality, tracking and image registration.

8.1 Image quality

Overall image quality of the rendered image is mostly dependant on the quality of the rendered 3D object, but quality of the camera image used as background is also important.

Rendering engine of the application is implemented in Wikitude framework. This engine is implemented in Javascript language, resulting in great multi-platform usage, however with very high CPU load for relatively basic renderer. Result image is far from photo-realistic, because of absenting photo-realistic techniques. Resolution of both the camera image and 3D model are also reduced to lower the performance demands.

Achieving the best possible augmented reality result would require also 3D object interference with camera image, and vice versa. For example dropping shadows to each other, using the real-world light sources in AR space etc.

Despite a lack of these features in application, quality of the image is sufficient for demonstration purpose of this thesis.

Performance of the application is easily measured by achieved frames per second (FPS) value of the rendering. On LG G3 this application performed around 30 fps with all of the tested 3D models. On the NVIDIA Shield tablet, result was much higher, performing around 60fps under the same circumstances.

8.2 Tracking

Tracking is a necessary process to achieve the Image registration. Tracking method used in this thesis application is one of the Non-visual tracking methods, particularly the Geo location. Device's sensors are used to determine camera location and position to render 3D object located on coordinates chosen by user.

Choosing the coordinates using Google Place Picker works flawlessly. Picked location is designated with more than sufficiently precise coordinates. Determining the elevation at the given coordinates using the Google elevation API is unfortunately not that precise. At some tested locations the measured elevation diverged up to 2 meters compared to real one. As a work around to partially solve this problem, user should turn off the GPS and

Glonass positioning to enter the low accuracy mode, where the elevation of the 3D model is set for the same level as the device is.

Accelerometer, digital compass and gyroscope combined made a very good result in the accuracy of the model placement and it's stability. Digital compass should be calibrated after longer time without using it, simply moving the device in a trajectory of number 8.

Location lock worked precisely on both devices in the open space areas with the clear sky view, where this lock was achieved under 3 seconds with sufficient accuracy. Location lock in urban areas or with the worse weather conditions can result in unwanted 3D model position glitches.

8.3 Image registration

Overlapping the rendered image on camera image is the final important step to achieve the Augmented reality. Rotation of the 3D object along with it scale should be set properly to fit into real scene.

Rotation around the z-axis is done by the animation triggered by touching the 3D model. Rotation around the other axes is not that necessary in most use cases, where the ground is relatively flat. To ensure correct functionality in other cases where the ground is steep, user should change these parameters when exporting the 3D model. Scale of the model should be also adjusted to real life units on its export.

Chapter 9

Conclusion

Last chapter is dedicated to discussion about achieved results and proposals to future improvement in the application and its utilization.

9.1 Achieved results

Aim of the application was to provide Augmented reality visualisation of buildings, mostly aimed to architectural field. Although the result image quality is not realistic, demonstration purposes of the application are fulfilled. In most common scenarios this application works well with few visual glitches. Limited battery life is a disadvantage, but it must be accepted as all the sensors and performance are used for the result.

9.2 Future improvements and utilization

First and the most apparent improvement that could be made to application of this kind, is image quality improvement. Mobile devices are still limited to provide real-time photo-realistic imagery, but with rendering engines implemented natively for targeted platforms, in help with software improvements such as Vulkan API, situation is getting better.

Another improvement should be made to tracking methods, while it still does not work 100% correctly. To achieve this, using only one technique is not enough, so combination of multiple tracking methods should be used, for example marker-less visual tracking by Wikitude.

Utilization of this application could be used to rendering also the interiors of buildings. This would have required very precise tracking technique, but it is not impossible. This usage would also be tempting to use in combination with virtual reality headsets to achieve more convincing results.

For Augmented reality as a concept itself, this application could help to widen usage of this concept with a huge potential, proposing more realisations of augmented reality applications in every day life.

Bibliography

[1] Alger, M.: Visual Design Methods for Virtual Reality. 2015.
Retrieved from:
http://aperturesciencellc.com/vr/VisualDesignMethodsforVR_MikeAlger.pdf

[2] Andrew I. Comport, I. E. M. M. I. M. P., Member; Franc¸ois Chaumette, I., Member: Real-Time Markerless Tracking for Augmented Reality: The Virtual Visual Servoing Framework. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*. vol. 12, no. 4. 2006.

[3] Brownlee, J.: Augmented Reality SDK Comparison [online].
Retrieved from:
http://socialcompare.com/en/comparison/augmented-reality-sdks

[4] Daniel Mathis, B.: *Zielgruppenorientierte mobile Augmented Reality Anwendung für die Ortsplanung am Beispiel der Gemeinde Hohenweiler*. Master's Thesis. Fachhochschule Vorarlberg. 2015.

[5] Erkan Bostanci, S. E., Nadia Kanwal; Clark, A. F.: User Tracking Methods for Augmented Reality. *International Journal of Computer Theory and Engineering,*. vol. 5, no. 1. 2013.

[6] GmbH, W.: About Wikitude SDK.
Retrieved from: http://www.wikitude.com/about/

[7] Hinum, K.: NVIDIA GeForce ULP K1 (Tegra K1).
Retrieved from: http:
//www.notebookcheck.net/NVIDIA-GeForce-ULP-K1-Tegra-K1.108453.0.html

[8] Inc, A.: FBX SDK Programmer's Guide.
Retrieved from:
http://docs.autodesk.com/FBX/2014/ENU/FBX-SDK-Documentation/index.html

[9] KHRONOS: Vulkan 1.0.17 - A Specification.
Retrieved from:
https://www.khronos.org/registry/vulkan/specs/1.0/pdf/vkspec.pdf

[10] Klein, G.: *Visual Tracking for Augmented Reality*. PhD. Thesis. University of Cambridge. 1 2006.

[11] Layar: About Layar.
Retrieved from: https://www.layar.com/about/

[12] Layar: Layar 3D Model Converter documentation.
Retrieved from: https://play.google.com/store/apps/details?id=com.layar

[13] Layar: Layar Android Application.
Retrieved from: https://play.google.com/store/apps/details?id=com.layar

[14] Lumera, J.: Why Retina Isn't Enough. 2012.
Retrieved from:
http://www.cultofmac.com/173702/why-retina-isnt-enough-feature/

[15] Lumera, J.: Is Augmented Reality the future of technical documentation. 2013.
Retrieved from:
http://www.tcworld.info/e-magazine/technical-communication/article/is-augmented-reality-the-future-of-technical-documentation/

[16] McHenry, K.; Bajcsy, P.: *An Overview of 3D Data Content, File Formats and Viewers*. National Center for Supercomputing Applications University of Illinois at Urbana-Champaign, Urbana, IL. 2008.

[17] Metaio: Product Support.
Retrieved from: https://www.metaio.com/product_support.html

[18] Multidots: Marker-based AR.
Retrieved from: http://www.multidots.com/augmented-reality/

[19] Nagy, M.: *Utilizing Building Information Models with Mobile Augmented Reality and Location-Based Services*. Master's Thesis. Norwegian University of Science and Technology. 2013.

[20] Nvidia: NVIDIA IRAY.
Retrieved from: http://www.nvidia.com/object/nvidia-iray.html

[21] Peter Shirley, S. M., Michael Ashikhmin: *Fundamentals of Computer Graphics, Third Edition*. A K Peters, Ltd.. 2010. ISBN 13-978-1568814698.

[22] phonearena: LG G3 vs Nvidia SHIELD [online].
Retrieved from: http://www.phonearena.com/phones/compare/Nvidia-SHIELD,LG-G3/phones/8347,8812

[23] Pirazda, U.: Khronos Group's 'Vulkan' API, Performance Numbers Revealed.
Retrieved from: http://wccftech.com/khronos-vulkan-api-performance-numbers-89-star-dust-benchmark/

[24] Project, W.: GP2Mark.
Retrieved from:
https://play.google.com/store/apps/details?id=com.wizapply.gp2mark

[25] Rao, G. S.: *Global Navigation Satellite Systems With Essentials of Satellite Communnications*. Tata McGraw Hill Education Private Limited, New Delhi. 2010. ISBN 13-978-0-07-070029-1.

[26] SlyBeaver: GPS info (plus GLONASS).
Retrieved from:
https://play.google.com/store/apps/details?id=ru.slybeaver.gpsinfo

[27] Tomas Akenine-Moller, N. H., Eric Haines: *Real-Time Rendering, Third Edition*. A K Peters, Ltd.. 2008. ISBN 13-978-1568814247.

[28] Wikipedia: Phong reflection model.
Retrieved from: https://en.wikipedia.org/wiki/Phong_reflection_model

[29] Wikipedia: Wavefront .obj file.
Retrieved from: https://en.wikipedia.org/wiki/Wavefront_.obj_file

[30] Wikitude: Wikitude 3D Encoder Documentation.
Retrieved from: http://www.wikitude.com/external/doc/documentation/latest/htmlcss/encoder.html#wikitude-3d-encoder

[31] Wikitude: The Wikitude App.
Retrieved from: http://www.wikitude.com/app/

[32] Zhanpeng Huang, P. H. C. P., Weikai Li: CloudRidAR: A Cloud-based Architecture for Mobile Augmented Reality. 2012.
Retrieved from:
http://www.cse.ust.hk/~panhui/papers/mars2014_cloudridar.pdf

Appendices

List of Appendices

Appendix A

CD content

CD directory structure

- **src** - directory containing the source code files of the application.
- **thesis** - directory containing source code of the thesis in LaTeX.
- **README.txt** - text file describing compilation of the application.
- **bachelor_thesis_xminda00.pdf** - text of the thesis.

Appendix B

Manual

Starting the application

- **Compiling the application**

  Directory `src` should be loaded as a project into Android studio version `2.1.2` or later. Then build the project and run it on a compatible Android device. Web server PHP script should be accessible from web, if you want to run it on a different server, just change the URL in the application.

- **Running the application**

  Controls and behavior of the application are described in Application Design chapter.