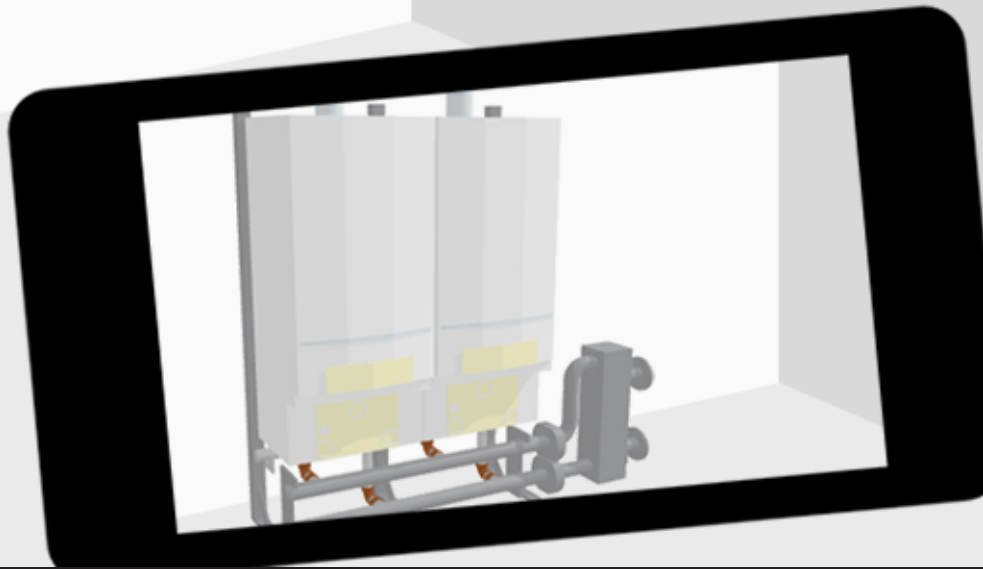
AR for Sustainable Renovation of Buildings

D.M. van den Berg
J.M. Helgers
M. Kralt
S. N. Lans

energie paleis



TU Delft

AR for Sustainable Renovation of Buildings

by

D.M. van den Berg
J.M. Helgers
M. Kralt
S. N. Lans

to obtain the degree of Bachelor of Science
at the Delft University of Technology,

Bachelor project coordinator:	Huijuan Wang, Otto Visser
TU coach:	Prof. Dr. Elmar Eisemann
Client adviser:	Chris Hendriks

TU Delft

Preface

This report concludes the TI3806 Bachelorproject, and contains all information concerning the 'AR for Sustainable Renovation of Building' project. 'During' this ten week project we've been commissioned by Energiepaleis to developed an Augmented Reality application that would give their clients a better 'insight' into, and understanding of their products.

We would like to thank (the people of) Energiepaleis and Chris Hendriks for their role as customer and for giving us the opportunity to work on this project. We would also like to thank Elmar Eisemann for coaching and assisting us for the duration of this project.

D.M. van den Berg
J.M. Helgers
M. Kralt
S. N. Lans
Delft, July 7th 2017

Contents

1

Summary

Energiepaleis, a company that realizes sustainable home renovations, requested an application that uses augmented reality to show their customers beforehand, what certain sustainable devices would look like in their homes after renovations. A marker is placed on the spot where the device needs to stand and then the application will show the device on the screen.

Before creating the application, research was done in the field of augmented reality. The research gave insights in the possibilities of augmented reality in mobile phone applications. The sustainable devices are represented by BIM models, these models are visualized using ARToolKit.

The implemented system can be divided in three parts, augmented reality platform, user interface and the core of the application. The augmented reality platform handles the rendering of the models. The user interacts with the user interface and the core of the application controls the rendering based on the user actions.

Testing the application is done in two different ways, manual and automated testing. Manual testing is done after each new feature is implemented. Automated testing is used to assure correct behavior.

2

Introduction

Augmented reality was first introduced in the 1960s [18], by this time a large head-mounted display was used to show the user simple images. Since 2013 augmented reality is available for consumers audiences, for instance manufacturers use it to assist in building or repairing their products. In the next year Google starts shipping the Google Glass, with this product they start a trend in augmented reality for consumers[9]. Currently, July 2017, user can use their mobile phone to place different kind of objects in the space or to play games that use augmented reality.

With this in mind, the company Energiepaleis [6] wants to have an application that uses augmented reality to give their customers an idea how their house will look like after a renovation. Energiepaleis specializes in sustainable renovations, so they especially want to show their customers sustainable products. Most people don't know how these products look like, which results in not buying them. Energiepaleis wants to change this, they want to show people how these sustainable products will look like in their house with our application and hope that this results in more people buying sustainable products.

2.1. Structure of this document

Before we started making the application we were in the research phase, in this phase we had to conduct research and define the actual problem. Appendix A shows the actual research that we conducted and in chapter 3 we define the problem that we had to solve. In chapter 4 we explain the implementation and the design of the application and describe the structure of the project. To verify that the code that we wrote are behaving in the correct way we wrote tests, this is explained in chapter 5. Next we will describe the process and the tools that we used in chapter 6. And finally we will give a conclusion, discuss the project and give recommendation in chapters 7 and 8.

3

A real-world problem

The main idea behind the BEP (Bachelor End Project) is that a team of students produces a product that is commissioned by a client and solves a real-world problem. The client is a real-world stakeholder, who commissions the team to develop a software solution to address a specific problem.

This project we've been commissioned by the people of Energiepaleis to produce an AR application that could help in the process of sustainable renovation of houses. For a more detailed description of the company, the research and problem analysis see the research paper A included as an appendix.

In short, Energiepaleis is a company that helps people renovate their homes sustainably. They deliver smart custom designs for their customers, based on their specific wishes. These designed solutions, and the products (think of: vents, pumps, heaters, etc.) that together comprise these solutions will be built into their customers homes. Since people care a lot about the Aesthetics of their houses, It could be really useful for them to be able to see what the renovation would look like beforehand. This is where we come in, the people of Energiepaleis asked us to create an Augmented reality application that does exactly that, show BIM (building information modeling) models in their customers houses.

3.1. Requirements

After deliberation with the customer and the TU-coach, the following requirements came to light.

3.1.1. Must haves

- BIM (Building Information Modeling) models representable in the AR-layer.
  I.e. A way to show BIM models in the AR application, be it by conversion or not.
- Ability to recognize 1 marker and add 1 object to that marker.
- Represented models are accurate in size compared to their surroundings.
- No specialized hardware (excluding Android phones and markers).
- Able to use a minimum of 10 different models (one of each type: pump, vent, etc.) that are present locally (on the device). Not necessarily simultaneously, but at least 10 different pre-converted models to give the user a good visualization of different types of products.
- Application functions in an inside environment (as opposed to outside).
- Able to represent objects of dimensions between (0.01 m x 0.01 m x 0.01 m) and (3.0 m x 3.0 m x 3.0 m).
- Software runs on Android version 4.0.3 onwards.

3.1.2. Should haves

- Real-time visualization of models. –as opposed to static image based AR.
- Ability to move objects around within AR environment.
- Ability to recognize more than one marker and add a different object to each marker.
- Ability to add models to be displayed. import new models to be used in the application.
- Energiepaleis logo used in markers.

3.1.3. Could haves

- Even more markers and objects.
- Playing sounds that imitate the actual devices' sounds.
- Dynamic lighting (Choose/detect direction of lighting, use multiple light sources.)
- Realistic models in terms of texture and color.
- Screen-shot function to capture an image of the model in the real environment for later use.
- Application functions in outside environment.
- Ability to choose markers of different sizes as defined by the user.
- Ability to choose different camera resolutions.

3.1.4. Won't haves

- Marker-less model placement.
- Measuring distances between objects or walls.
- Possibility to make parts of the models translucent so that the inside can be viewed.
- Possibility to make parts of the real environment translucent, so that models can be represented within/in place of the real objects.

3.2. Use Cases

the following use cases were generated after deliberation with the client:

3.2.1. Selecting a 3D-model

Actor	System user
Precondition	The user has just opened the application and is currently in the main menu.
Postcondition	A 3D-model is present on the physical marker that the user assigned.
Actions	1. The user presses the 'select model' button and enters the model selection menu. 2. The user clicks on the marker that he wants to assign a model to. 3. The user clicks on the 3D-model image that he wants to place on the previously selected marker. 4. The user returns to the main menu by pressing the "back" button on the device. 5. The user enters the camera view by pressing the "go to camera" button. 6. The user aims the camera at the physical marker that he assigned a model to and can see the 3D model on the location and orientation of the marker.

3.2.2. Visually comparing sustainable devices

Actor	System user
Precondition	The user has just opened the application and is currently in the main menu.
Postcondition	Multiple sustainable devices are seemingly present in the users' environment.
Actions	1. The user executes the use case 'Selecting a 3D-model'. Steps 2 and 3 are executed at least 2 times in a row. 2. The user places the physical markers that were assigned a model next to each other on the location where the devices should be placed in case they were actually present real-life. 3. The user aims the camera at the physical markers that he just placed and can compare the real-sized models of the sustainable device.

3.2.3. Importing a model

Actor	System user
Precondition	The user has just opened the application and is currently in the main menu.
Postcondition	New models are present in the model selection menu for the user to choose from
Actions	1. The user presses the 'import object file' button and enters a file browser. 2. The user chooses the model he wants to import and clicks on it, after which the model gets imported.

3.2.4. Translating or Rotating a model

Actor	System user
Precondition	The user has just opened the application and is currently in the main menu.
Postcondition	model is translated relative to the marker
Actions	1. The user executes the use case 'Selecting a 3D-model'. 2. The user aims the camera at a single marker (the one that is connected to the object he wants to translate). 3. The user presses the 'move' button and 6 arrow buttons appear 4. The user chooses the direction he wants to move the model in and presses that arrow. 5. After the user is done he presses the check mark to finish.

3.2.5. Resizing marker

Actor	System user
Precondition	The user has just opened the application and is currently in the main menu.
Postcondition	marker of user specified size can be used
Actions	1. The user presses the 'select model' button and enters the model selection menu. 2. The user long clicks (holds click for 2+ seconds) the marker that he wants to resize, until a number picker pops up. 3. The user chooses the size (in cm) of the marker and presses 'OK'. The marker has now been resized and can be used in the camera view.

4

Design and implementation

In the previous chapters, the problem that the application solves has been described. The requirements have also been specified and insight into the actual usage of the application has been provided by use cases. This chapter builds upon the overview of the previous chapters to provide the overall design of the application. First, the high-level design will be given, consisting of the different system components. After these individual components have been identified, a more detailed overview of each component is provided. Implementation details are provided for the most relevant aspects. Then, the design of the physical markers that are used is discussed. After that the usage of IFC will be explained. At last, a description is provided of the supported devices and the implications of porting the application to non-supported devices.

4.1. High-level design

The system can be split up into three distinctive components:

1. An Augmented Reality platform that provides tracking of the locations where 3D-models should be placed. It also provides the functionality to display those 3D-models overlayed on a camera view.
2. The user interface that is visible to the user. It communicates the actions of the user to the application code.
3. The core application code. It takes the actions of the user from the user interface and controls the AR platform based on those actions.

A sequence diagram of these three components in a typical scenario is presented in figure 4.1.

4.2. Component design

4.2.1. Augmented Reality platform

The AR platform of choice is ARToolKit, as argued in the research paper. ARToolKit consists of C and C++ libraries, which implies the usage of the Android JNI Framework as a bridge between Java and C/C++ code. A version of ARToolKit is available for Android, including both a Java library (ARBaseLib) and a native C++ wrapper library (ARToolKitWrapper). Using both of these libraries will mean less native development and a lower complexity for the developers, but comes at the cost of reduced control over the ARToolKit functions. Alternatively, only the C++ wrapper can be used, or only the core ARToolKit libraries. For our application, we have decided to use both the ARBaseLib and ARToolKitWrapper libraries, because the features provided by these libraries cover our needs. The overall structure of ARToolKit, using both of these libraries, is depicted in figure 4.2.

In our application, we have chosen to supply our own renderer to ARBaseLib so that we are able to use the C++ library Eden, which can import Wavefront .obj files and is easily integrated into the functionality of ARToolKit. Additionally, creating our own renderer gives us more granular control over the scene that's being rendered as it gives us the freedom to set up and change OpenGL parameters as

Figure 4.1: Sequence diagram of the high-level design in a typical use case.

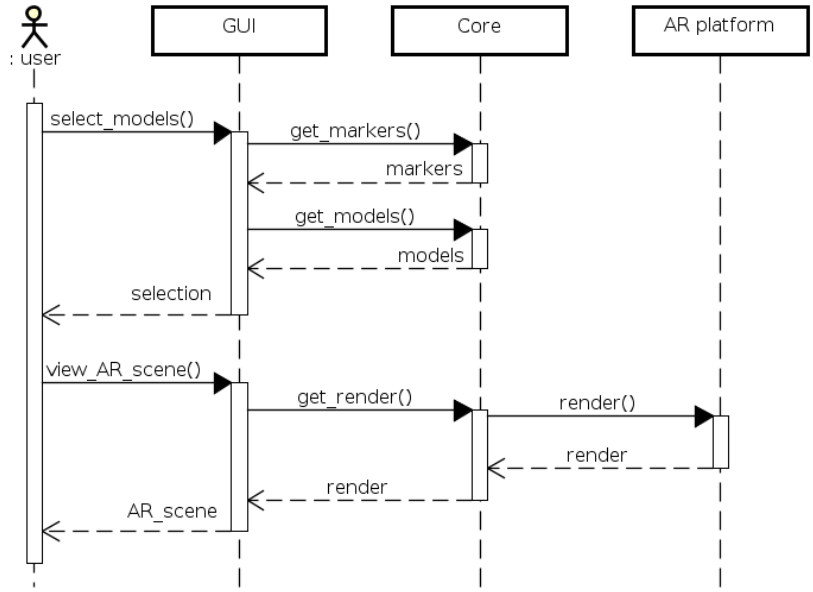
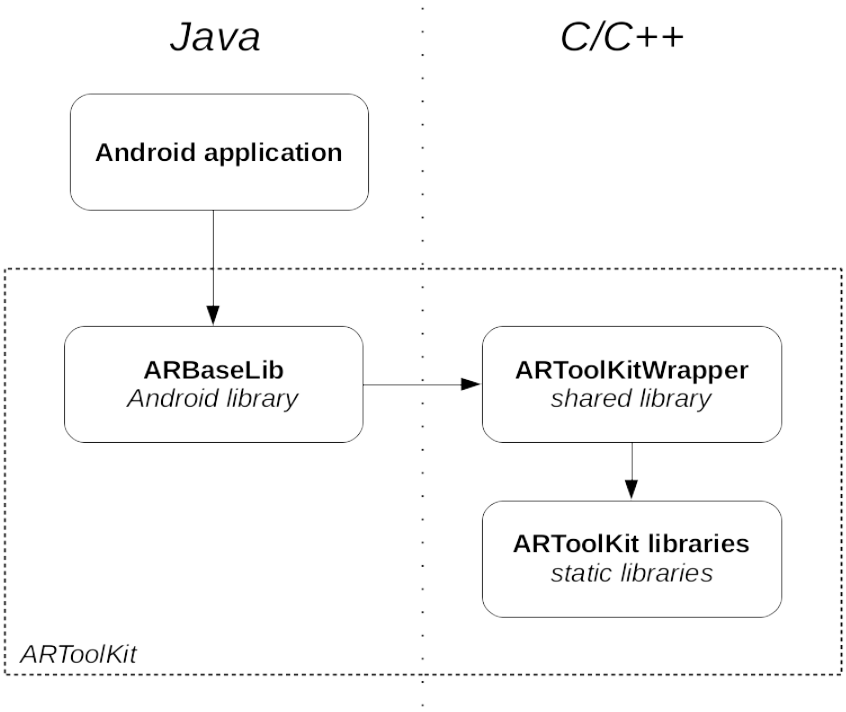


Figure 4.2: Structure of ARToolkit in the application.



we see fit. Supplying our own renderer involves extending the ARRenderer class from ARBaseLib and creating our own C++ file that handles loading the correct models using Eden and processes the frames using other ARToolKit libraries. The extended ARRenderer class can then use the native functions from the C++ file to render the final result.

4.2.2. Core application code

The core application code can be broadly split up into two parts:

1. The renderer that is supplied to ARBaseLib and the native C++ code that it uses, as discussed

in the previous subsection.

2. An utility package containing the data classes that the application uses and a resource manager that takes care of reading and writing resources from/to the device.

Through the NativeRenderer class, the data required for the application to draw the models is supplied to the native code. This consists of the selected markers, the associated models and the marker size. With this information, the native code prepares the selected models for rendering through the methods provided by Eden. ARToolKit keeps track of the markers, and their visibility is queried for every frame that the renderer receives from the camera. When a marker is visible, the associated model is drawn.

The 3D models that our application mainly uses are programmatically converted through IfcConvert. Unfortunately this process is limited in its accuracy, resulting in surface normals that are not always facing outwards. In order to represent the models with reasonable accuracy, the 3D scene for our renderer is configured to render double sided faces. In order to ensure proper visibility of the models, the scene is lit with a combination of ambient lighting, and a diffuse light that always points in the same direction as the camera.

The data classes of the utility package consist of a Marker and a Model class. Both of these classes contain basic information such as the name of the marker/model, the location of the corresponding picture and the path of the marker/model file on the device. They contain all information that is needed in the GUI to display them to the user and the information that is needed by ARToolKit to load the actual markers and models.

The ResourceManager class is responsible for creating Marker and Model instances for all available markers and models. In order to do this, the directories containing the marker and model files are scanned. After this is done, the ResourceManager makes all Marker and Model instances available for use by the other parts of the application. The ResourceManager is also used to import additional model files that were selected by the user. These files can be either in BIM format (.ifc) or in the Wavefront object format (.obj). Since the the same set of models and markers is used throughout the application, only one instance of the ResourceManager should exist. To ensure this, the singleton design pattern is applied. Figure 4.3 provides a class diagram of the Marker, Model and ResourceManager classes.

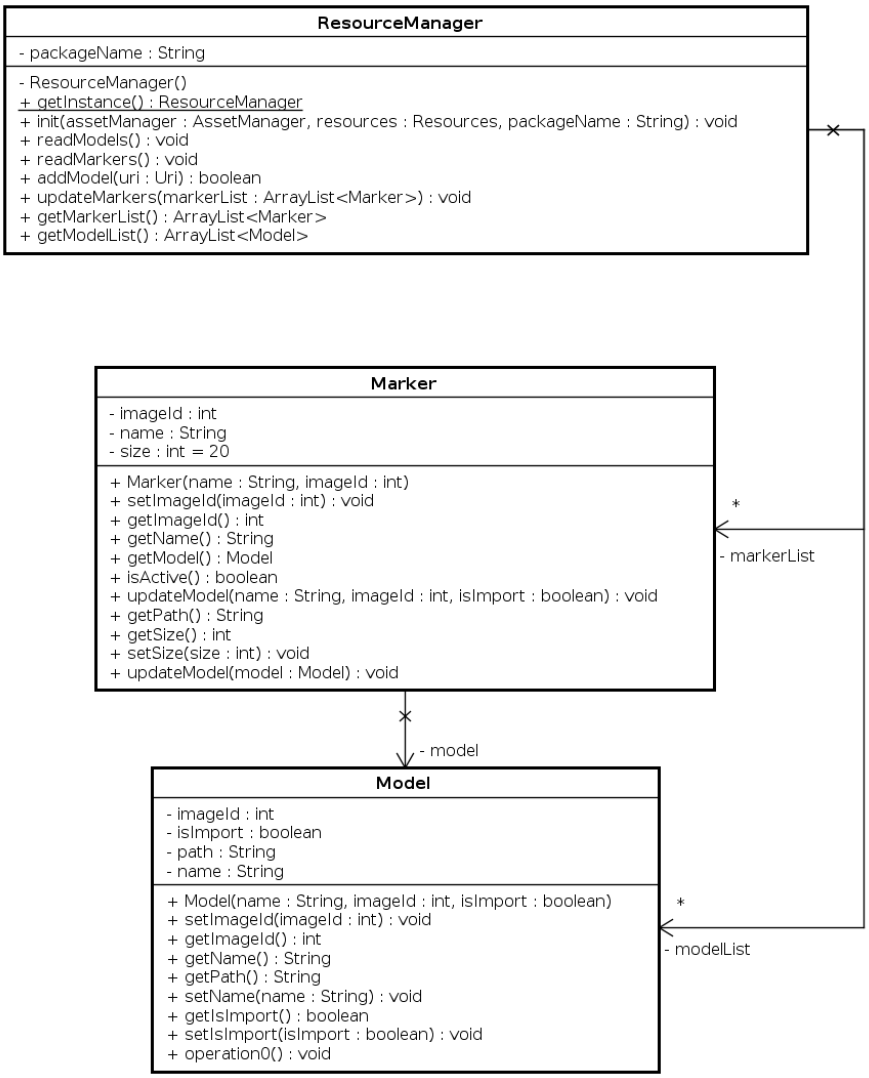
4.2.3. Graphical User Interface

On Android, each separate screen of the application corresponds to a class extending from the Activity class. The Activity provides the window that is presented to the user and functions as the entry point for the interaction of the application with the user [1]. Additionally, an XML file can be used for each Activity to define the layout and styling of the window. In our application, we use an XML file for each Activity in order to separate the view that is presented to the user from the code behind it. However, the view is not separated from the code in the same way as in the Model-view-controller (MVC) pattern. In this pattern, the underlying data model is updated by a controller that is operated by the user. The model then updates the view and the user can see the updated view. This pattern implies that the controller does not have access to the view. On Android, where the Activity acts as controller, the controller does have access to the view. However, this is simply the way in which Google chose to implement the Android user interface and it does not impact the user experience in any way.

In order for the application to work with ARToolKit, the Activity that presents the augmented camera view to the user needs to extend the ARActivity class of ARToolKit. Also, an Android FrameLayout must be provided, which is a stacked container for GUI elements. ARToolKit can then place both a camera view and a 3D view of rendered objects into the FrameLayout. The FrameLayout makes both layers visible on top of each other, achieving the desired Augmented Reality view. Figure 4.4 illustrates this behavior.

The styling for the GUI is inspired by the website of the client [6]: the same color scheme is used throughout the application.

Figure 4.3: Class diagram of the util package.



4.3. Marker design

AR applications allow virtual imagery to be superimposed on live video. This feat is achieved through the use of a marker, a black square marker. A square marker is made up of a dark black border and a light colored center (usually white) embedded with a high contrast image referred to as a pattern, as shown in figure 4.5. The pattern is what makes a square marker unique. Patterns are used in calculating the orientation of a square marker, thats why patterns need to be rotationally asymmetric.

Multiple markers can be used to represent different objects in an AR application. However, using patterns inside a marker comes at a computational cost, since the pattern in the image captured from the camera must be compared against every known marker pattern at all four possible orientations. For a handful of markers, this computation comes at small cost, but as the number of markers rises the cost becomes significant. Additionally, pattern based markers are more likely to be confused by the tracker, since there is a lower degree of uniqueness within the set of markers, leading to markers being misrepresented as each other.

A solution to the tracking difficulties, caused by the use of multiple markers, is to use so-called two-dimensional barcode markers, they have a predetermined pattern of black and white squares in a grid arrangement and offer the possibility of error detection and correction. Using better error detection and correction results in a smaller set of markers being available, but lower likelihood of markers being mistakenly recognized during tracking.

Figure 4.4: Combining the AR and camera layers in a FrameLayout. Picture courtesy of [4].

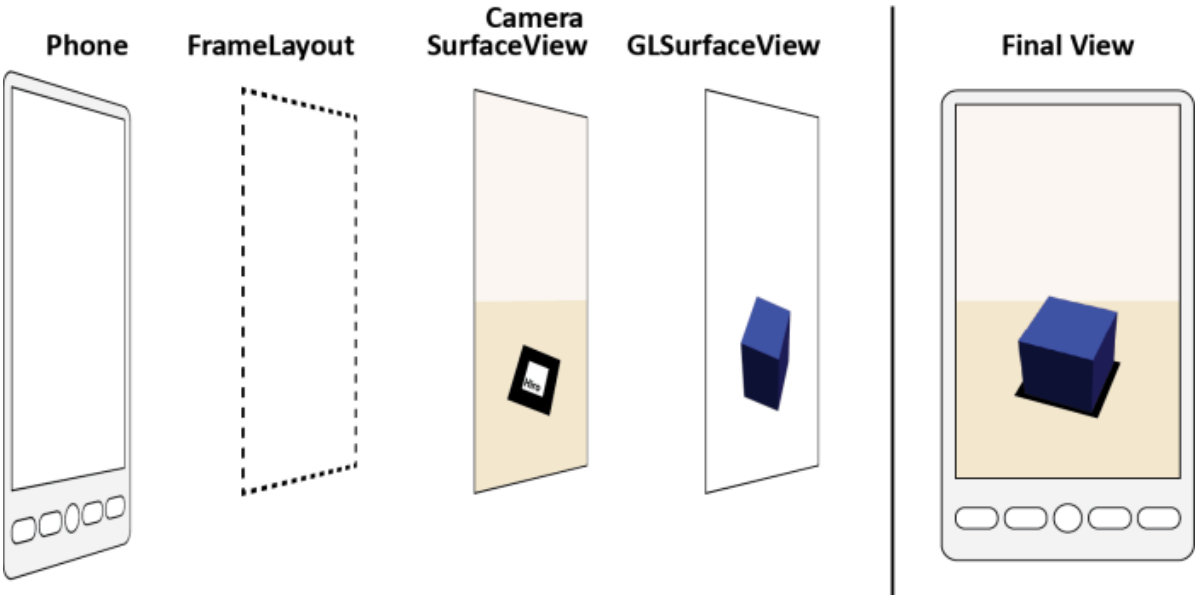


Figure 4.5: A pattern marker.

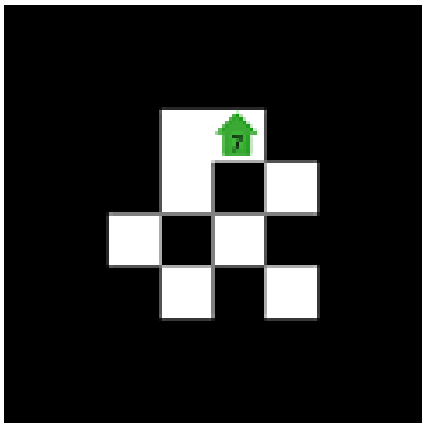


Figure 4.6: A matrix marker.

In our case a 4x4 grid was used in conjunction with BCH codes (13-5-5, Bose–Chaudhuri–Hocquenghem codes, a class of cyclic error-correcting codes), shown in figure 4.6. This yielded a total number of 32 markers with hamming distance of 5, a relatively large hamming distance and the possibility to add markers at a later date.

4.4. IFC

As explained in the research paper, the ability to use BIM models in our application means that we should be able to import models in the IFC format. Because of the complexity of the IFC standard and the availability of existing IFC loaders/importers, we decided that we would use one of those existing implementations. Because we read in the models in C/C++, Java based IFC readers such as the IFC Tools Project [11] were not suitable. The most suitable option turned out to be IfcOpenShell [10], which contains the executable IfcConvert. This executable converts IFC files into Wavefront .obj files and material files (.mtl), which can then be read by using the Eden library as discussed in subsection 4.2.1. Calling the executable from within the application can take some time, ranging from a few seconds to 30 seconds for the model files that were tested. Because of that, IfcConvert is called in a separate thread, so that the main thread remains free to display a loading pop-up to the user.

4.5. Compatibility

The application is targeted at Android versions 4.0.3 (API level 15) and above, since that yields a coverage of nearly all Android devices in use today [3]. The C/C++ code and libraries that are used need to be compiled for each separate architecture, which limits the compatibility to devices for which the C/C++ code is compiled. However, the C/C++ code is compiled for all possible architectures that run Android, which means that the support is not limited to specific architectures, but only to specific Android versions (4.0.3+).

For each new Android version and corresponding API, an increasing amount of functionality becomes available for developers to use. These newly added functionalities are only available starting from the API level in which they were added and can not be used on older devices. However, a support library is available that makes newly added functionalities available for older devices as well [2]. In order to use the Material Design styling of the GUI, that was added in Android 5.0, we use this support library. As a result, the application is up-to-date in terms of its looks, without compromising the compatibility with older devices.

Most mobile devices run either Android or iOS. Since developing for iOS would require access to a system running macOS, which none of the team members had, we were not able to develop for iOS. ARToolKit is available for iOS as well, so changing the Augmented Reality platform would not be necessary when developing for iOS. However, adding support for iOS would involve redesigning the other parts of the application.

5

Testing and Validation

Testing is one of the important aspects of our project, because you want to verify that the application behaves in a proper manner. This is verified with two different types of tests, manual testing and automated testing. Section 5.1 goes about the manual tests that are executed and section 5.2 explains the different automated tests that are used in this project. Finally the quality of the automated tests is ensured in section 5.3.

5.1. Manual testing

Manual testing consists of two parts. The first part is just running the application, on a mobile device or an emulator on your computer, and see if the application works in the correct way, for instance check if the correct screen is opened when you press a button. The second part is running the application in debug mode, this makes it possible to for us to see the actual values of the variables that are used in the application, an example of this is to check whether a model is added to a marker when the user selects a model.

These two parts of manual testing are closely related. First the tester just run the application, part one, and when everything works as expected he stops manual testing. But when something behaves in a way that was not intended, then the tester will run the application in debug mode, part two.

The tester will perform the first part of manual testing every time a new feature for the application is implemented, these features are defined in section 3.1. To test the new features we defined use cases in section 3.2, these use cases tell the tester the exact steps he should take to see if the feature works properly. Figure 5.2 shows the first three steps performed by the tester of the use case in section 3.2.1, in this figure the red dots are the clicks that are done.

As explained before, when something goes wrong in manual testing the application, the tester will use the debug mode to precisely find out where the mistake is made. When the tester know that a certain model should be shown in the list when a marker is selected, but that model is not shown. He can stop the application when that part of the code is reached and check see what the values of the variables are. In that way he can check what goes wrong, figure 5.1 shows that there should be a model present with the name 'combiboiler'.

Figure 5.1: Debugging the select marker screen.

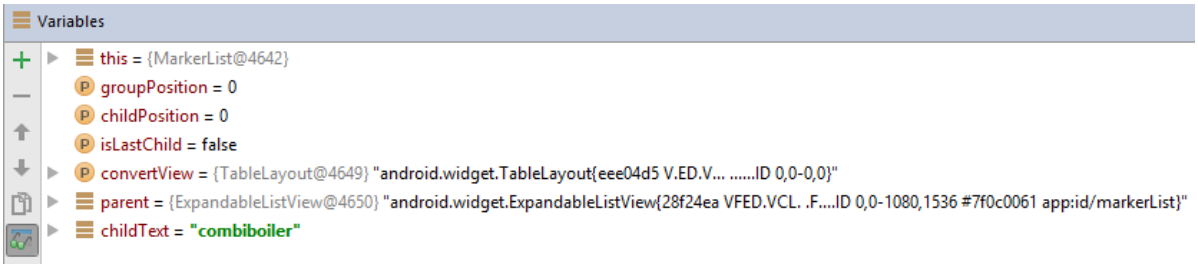
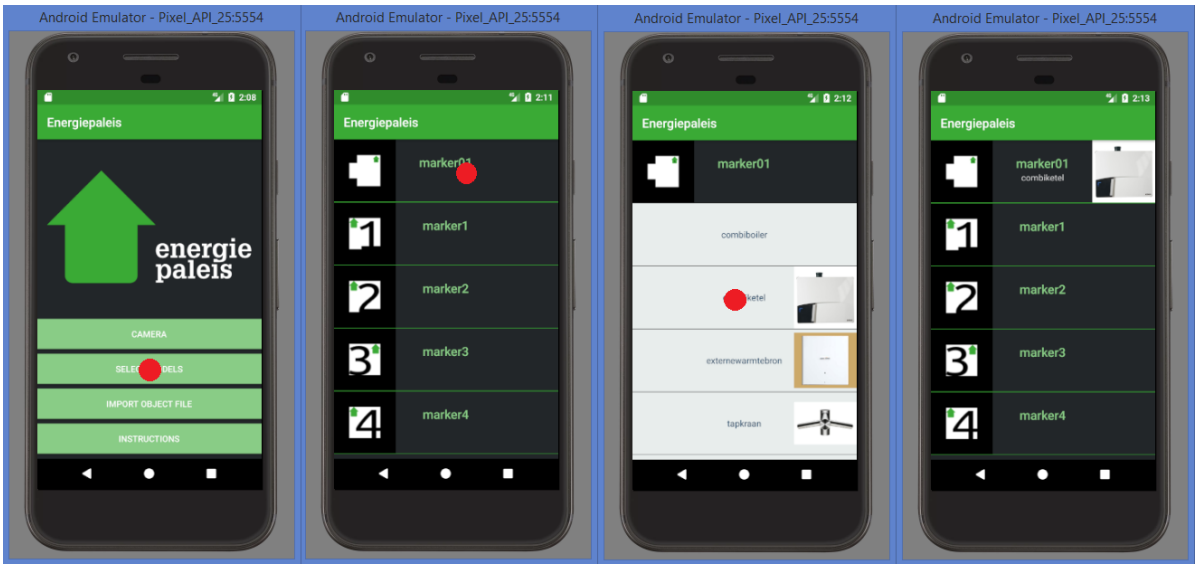


Figure 5.2: First steps of use case 3.3.1.



5.2. Automated testing

In this section we will explain how our project is tested with automated, self written, tests. Automated test are an important aspect of this project, because they can be used to verify the correct behavior of individual components of the application. They also provide a way to ensure that previous functionalities continue to work when adding new features to the application or enhancing existing features.

Because our project is an application for the mobile phone and we use C++ to draw the models in the application, there are three different types of code that need to be tested.

- Code that stores and computes the variables that are used.
- Code that provides the communication between Java and C++
- Code that is related to the user interface.

The first type of code is located in the util package **??**, and this type of code can be tested easily with the junit tests form Java. To test these classes simple tests are written that use mockito [12] to get the correct behaviour.

The other two types are harder to test. To test code that handles in- and output of the user interface there need to be some connection to a mobile device or emulator. And to test the code that communicates between Java and C++, you also need an device or emulator that compiles the C++ code.

Currently there are two possible ways to test these two types of code.

The first one is to write junit tests for the code and then run these tests with the robolectric[15] framework. This framework makes it possible to simulate an emulator, this means that the tester does not have to worry about running an emulator or connecting his device. The downfall is that this framework is not supported by sdk versions higher than 21.

The second possibility is to write androidTests for the code, to run these tests the framework espresso[7] is needed. But to run these tests the tester need to set up an emulator or connect an device.

We wanted to ensure that the sdk version of our application was as high as possible, so we choose to test our project using the espresso framework.

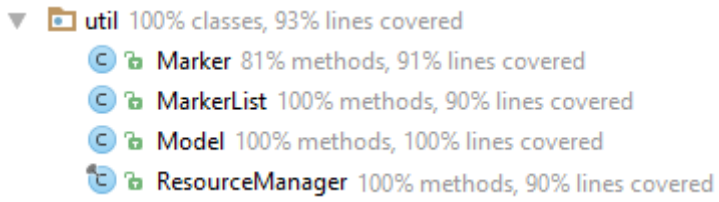
To test the activities with the espresso framework you first have to set up a testrule, with this rule you make sure that the activity will start up. Then you can check whether all the text and images are shown, and if the buttons perform the actions they are supposed to perform.

5.3. Validation

To verify that the code was tested in a proper manner, the test coverage percentage was measured. With this percentage you can see which portion of the code is tested. For the AndroidTests we were not able to get the test coverage percentage, this means that we can only give the coverage percentage of the tests that are written in junit.

Android Studio has a feature that makes it easy for the tester to get the test coverage for the junit tests. This is done by clicking on the tests and run them with coverage. After running the tests with coverage Android Studio states the percentage of code that is tested in the package manager. Figure 5.3 shows the coverage of these tests.

Figure 5.3: Coverage of the util package.



6 Process

This chapter gives a brief overview of the development process. The development tools, techniques and methodology are discussed in the coming sections. As well as a short reflection on the process (by each team member) in the last section.

6.1. Methodology

Given the requirements and request of our customer, the short time span of the project and the knowledge we gathered during the research phase, we knew that an agile development methodology was the way to go. Agile software development uses the least amount of documentation necessary to deliver a high quality project in relative short amounts of time. This however does not mean that an agile development methodology does not require documentation, it is simply kept to a minimum, not to unnecessarily burden a software development team. Agile encourages continues planning that involves the entire team. Changes can be made and features can be added throughout the duration of the project, with little to no ceremony.

A barebone version of Scrum [16] was chosen, with as little overhead as we could manage. The project was divided into sprints of about one or two weeks each. And almost daily meetings were held. The following development resources were used to help us in this:

6.1.1. Nuclino

”Nuclino is a unified workspace for teams. Bring all your team’s notes, tasks, and docs together in one place and collaborate in real-time.” [13] We have used Nuclino for all our documentation type needs during the research phase of the project. For example: notes of meetings, high level software descriptions, communication of ideas, feature descriptions and use cases. It was also used as a way to keep track of all the useful information found by team members relating to the project, be it theory around AR or useful code one might use at a later date.

6.1.2. Git & Github

We used Github’s [8] git implementation as version control for our software. Additionally we used the project management tools on the platform to maintain a backlog for our project. It is comprised of an ordered list of requirements for the product, consisting of features, bug fixes and other non-functional requirements to keep track of and assign work.

The general workflow within git was fairly standard. We strived to keep the master branch functional at all times. For every feature that was being implemented, a new branch was created in which development of that features took place. Once a feature was finished, a pull request was created which was merged after review by the rest of the team.

6.1.3. Android Studio

Android development was also a first for all of us. However, we were all familiar with Java, and some of the group members had experience with other Jetbrains IDEs. As a result, there was not a massive barrier to overcome in regards to getting started with implementing the application. We did come

across some problems with the integrated build tool of Android Studio, Gradle. This was mainly due to the usage of the ndk-build system to compile the C/C++ code before building the rest of the Java application. Using ndk-build required the experimental Gradle version instead of the regular one. Since each Gradle version requires a matching Gradle plugin, the correct plugin had to be installed for the compilation to work. Also, different Gradle versions can have a different syntax for the configuration files. This resulted in quite some research to find the correct Gradle plugin version and configuration file syntax in order to automatically compile the C/C++ code whenever the application was built. Once Gradle was successfully set up, no more issues arose with its usage.

6.1.4. C++

The members of the team all had limited experience working with C++. The use of this language was necessitated by our requirement to use existing models in combination with AR Toolkit. We had already obtained extensive experience with object-oriented programming languages through the courses provided by the TU Delft. Additionally, as there was some familiarity with OpenGL rendering, which is the main thing that needed to be implemented in C++, this did not cause problems in the long term.

6.1.5. Software Improvement Group

During the development process there was a single instance where we were required to send our code to the Software Improvement Group (SIG) in order to have it evaluated. The results of this evaluation can be found in C.1. As the feedback from this evaluation was very positive, we did not change need to adjust our methodology much for the remainder of the project. We did take their advice to heart and put forth a greater effort to implement more tests.

7

Conclusion

This project, we have been commissioned by Energiepaleis to produce an AR application that could help in the process of sustainable renovation of houses. Energiepaleis asked us for a way to show BIM (building information modeling) models in their customers houses using augmented reality. We worked on this for a couple months and have created a prototype after having researched the subject. The prototype incorporates all the must and should have requirements, and even some of the could haves.

Developing an AR application as we have done, is do-able in the relative short amount of time that we had, but if you really want to make something special, more time and some dedicated hardware are required.

To conclude, we enjoyed working together in this way, learning new things and creating something from scratch.

8

Discussion and Recommendations

In this chapter, we look back at decisions made during the project and the problems that we ran into during the project. An overall evaluation of the final product is also provided. The recommendations section provides an insight into future work regarding our application and Augmented Reality in general.

8.1. Discussion

8.1.1. BIM

Using BIM models results in the availability of actual products as models. However, the IFC format is primarily intended as a way to accurately describe properties such as dimensions and project information. It is not in the first place a 3D format. Thus, the models in the application will not be the best looking 3D models in existence. Also, because IFC is such a wide standard, one aspect of the geometry could be defined in multiple equivalent ways. When converting IFC to .obj, this could result in 3D models that are incorrect in some aspects, such as normals that face the wrong way. However, with BIM and IFC becoming increasingly popular, these problems will probably be resolved in the future.

8.1.2. ARToolKit

The extensive documentation that was available for ARToolKit was one of the reasons that we decided to use ARToolKit in our application. However, during the implementation phase, parts of the functionality turned out to be documented very scarcely or not at all. The most notable example of this was the documentation about the matrix markers and their usage in an application. Limited documentation is available stating that matrix markers are recognized differently than regular markers, but no further explanation as to its usage or examples were present. This left us unable to use these matrix markers in the way described in the documentation. In the end, we used the matrix markers as regular markers (by calibrating them in the same way). This might have negative effects on recognizability or traceability. However, the basic principle behind the matrix markers should still hold when calibrating them like a normal marker.

8.1.3. IfcOpenShell

IfcOpenShell had to be compiled for all processor architectures that we wanted the application to work on. This meant that the dependencies of IfcOpenShell also needed to be compiled for those architectures. These dependencies were Boost [5] and OpenCascade [14]. Compiling Boost was relatively easy, because Boost comes with its own build system, which could directly use Android standalone toolchains [17] that can be created with the Android NDK. OpenCascade uses CMake as its default build system, which, after some research, could also be used to successfully compile OpenCascade for the correct architectures. Unfortunately, compiling IfcOpenShell turned out to be more challenging. IfcOpenShell uses CMake as its default build system, but trying to invoke CMake resulted in compile errors that could not be solved by any of the team members due to a lack of experience with the compilation of C/C++ code. Luckily, some experience with the ndk-build system of the Android NDK followed from setting up the original project. As such, IfcOpenShell could eventually be successfully compiled for all Android architectures using the ndk-build system.

In the ideal case, IfcConvert would be present as a library instead of an executable. That way, it can be called from within the C/C++ code of the application, resulting in a cleaner application structure. Currently, the executable and the libraries it depends on are copied into the build folder of ndk-build, to ensure that it is included in the .apk file that is installed on the device.

8.1.4. Ethics

An Augmented Reality application like the one we have developed also brings some ethical considerations with its usage. For example in the case of our application there is the possibility that the 3D representations of the products aren't accurate to real life. Manufacturers have the ability to create and distribute 3D models of their products, so there is a possibility that these models will be more aesthetically pleasing than the actual product. This would constitute misinformation, which would harm the ability of a consumer to make a well-educated decision.

Privacy and security are also a concern when it comes to augmented reality applications. The most common use case for our application would be for user to utilize the app in their own home. If in a future iteration of the application the footage made by the app were to be transmitted for any reason, it would be sensitive to privacy infringement. If this footage is not handled with the appropriate care and security, it could potentially be intercepted by a third party.

8.2. Recommendations

8.2.1. ARToolKit 6

During development a new major release of ARToolKit went into open beta, ARToolKit 6. If our application were to be developed further beyond the scope of this project, then it would be wise to port the project over to this new version in order to allow for the utilization of new features and bugfixes.

8.2.2. Better 3D Models

The current workflow of our application consists of converting BIM models in the IFC format to Wavefront objects in order to render them. This severely limits the presentation of the models as BIM models are generally not created with the purpose of high quality renders. Also the conversion process is not flawless, which made us have to compromise on quality by sacrificing smooth lighting. With custom tailored models or a different workflow to ensure the quality of models, the visual presentation can therefore be improved greatly.

8.2.3. Markerless AR

In order to make the application function in a wider array of situations, the implementation of markerless AR is a consideration. This would require more complicated real-time image processing, the most basic of which would be surface detection. This would allow the placement of a model on a surface without requiring a marker. The biggest challenge for the Energiepaleis app that would need to be overcome is that of scale. Markers provide a very convenient way project markers at the correct scale, so an entirely different solution would be required for a markerless solution.

8.2.4. Improved Hardware

The introduction of new hardware with 3D cameras also influences the future possibilities of our application. When 3D cameras and depth-measurements become available on mainstream smartphones, those technologies could be applied to percept depth and size without markers. Other devices such as AR headsets could also become wider available, resulting in completely new platforms that our app could be developed for. Depending on the hardware and operating systems of these future devices, the app will most likely need to be redeveloped for those platforms, in order to use the features that are currently still unavailable on regular smartphones.

8.3. Evaluation of the final product

To evaluate the final product we check if all the requirements, defined in 3.1, are met.

First we evaluate the must haves that were created. The application can display at least 10 different BIM models. These models can be of different sizes and all these models will be displayed with their

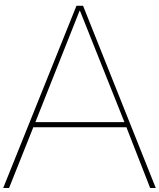
correct size. To run the application you mobile phone needs to have at least Android version 4.0.3 and no other hardware is needed. So all the must haves are met.

Secondly we see if the should haves are met. The models are placed on the screen in real time and the user can walk around marker to see different sides of the model. The user can select which marker, with the Energiepaleis logo, he wants to use and also select which model should be shown on that marker. If the user has more models that are not presented in the application, it is possible to import new models, and these imported models can also be shown. Thus also all the should haves are met.

Next we look at the could haves. There are multiple markers of different sizes available in the application, and because of the import function there is a wide range of models available. If there is a mtl file presented with model, the color and textures are shown, also the lightning is modified so that the models are clearly visible. It is also possible to use the application in the outside environment and the user can change the camera resolution.

Not all the could haves are implemented in the application, the application cannot play sounds. All of the won't haves also aren't met.

All the important requirements are met, must and should haves, and even a few less important ones are met, could haves. So the the final product can be used by Energiepaleis in a proper manner.



Research Report

Included on the following pages is the research report that we created as part of the research phase of this project.

Research Paper on AR For Sustainable Renovation

Daniël van den Berg 4361997
Marcel Kralt 4150767
Jeffrey Helgers 4318749
Sonny Lans 4007980

I. Introduction

This research report summarizes the results of the research phase for the bachelor project: "Augmented Reality for sustainable renovations of buildings". It will discuss the challenges in the current situation and reveal the optimal approach to reach a solution, after eliminating alternative relevant solutions.

It starts with a description of the project in section II and the requirements in section III. In section IV a brief overview of the involved technologies is given and in section V the relevant AR platforms are discussed. Ending with a Plan of Approach in section VI, the conclusion and discussion in section VII and the references.

II. Project Description

In this section we will describe the problem that is encountered by the company Energiepaleis. After this description we will describe the optimal solution for the company, as well as the solution that we think is realistically achievable within the duration of the project.

A. The current situation and its problems

When people are living in the same house for a long time or when they buy a house that is rather old, they want to renovate their house. Today the energy renewability of a household is of great importance to society, as well as to people individually. Especially when it comes to renewable renovation the general public is still uninformed, in addition to the higher costs and complexity of such renovations in general, it makes the process complicated and expensive. First people have to talk to a constructor, he comes to the house and decides if the plans of the owners are possible, then an architect is consulted to make sketches of the plans. And if these sketches are approved by the owners, the building phase can start. In the building phase there are a lot of different parties present that all have their own tasks. For instance glass setters and construction workers are needed in the renovations. They all have to work together to meet the demands of the homeowners.

1) Our proposed solution: Given the available time, hardware and expertise, we are going to be basing our solution around the AR visualization of solutions inside a client's home. Regular cellphone cameras provide enough visual acuity that this can be implemented as a mobile application. The app will be able to show a 3D representation of an installation imposed onto a real-life image. The goal is for this representation to provide an accurate representation of how it would fit into a home, giving consumers a convenient way of comparing different solutions for their renewable renovation. To simplify this process the application will use marker detection, and the scale of the objects will be based on the marker to obtain an accurate size relative to the surroundings. Our version of the application will be used by the employees of Energiepaleis, allowing them to give a preview of their proposed solutions to the customer without needing to involve any third parties.

III. Requirements

In this section, the requirements of our application are formulated based on the client's needs. They are structured according to the MoSCoW method, which is suitable for the Scrum software development method that we will be using.

A. Must haves

- BIM (Building Information Modeling) models representable in the AR-layer.
- Ability to recognize 1 marker and add 1 object to that marker.
- Represented models are accurate in size compared to their surroundings.
- No specialized hardware (excluding Android phones and markers).
- Able to use a minimum of 10 different models (one of each type: pump, vent, etc.) that are present locally (on the device).
- Application functions in an inside environment (as opposed to outside).
- Able to represent objects of dimensions between (0.01 m x 0.01 m x 0.01 m) and (3.0 m x 3.0 m x 3.0 m).
- Software runs on Android version 4.0.0 and newer.

B. Should haves

- Real-time visualization of models.
- Ability to move objects around within AR environment.
- Ability to recognize more than one marker and add a different object to each marker.
- Ability to add models to be displayed.
- Energiepaleis logo used in markers.

C. Could haves

- Even more markers and objects.
- Playing sounds that imitate the actual devices' sounds.
- Dynamic lighting (Choose/detect direction of lighting).
- Realistic models in terms of texture and color.
- Screenshot function to capture an image of the model in the real environment for later use.
- Application functions in outside environment.

D. Won't haves

- Markerless model placement.
- Measuring distances between objects or walls.
- Possibility to make parts of the models opaque so that the inside can be viewed.
- Possibility to make parts of the real environment opaque, so that models can be represented within/in place of the real objects.

IV. Technologies

A. Augmented Reality

Augmented Reality aims to extend the perception of the real world with a virtual layer. This virtual layer can contain different types of information and can be delivered in different ways. One of the common definitions for Augmented Reality [1] defines an AR system by three characteristics:

- Combines real and virtual.
- Interactive in real time.
- Registered in 3D.

Since the introduction of modern devices such as smartphones and Google Glass, Augmented Reality is receiving an increasing amount of attention. However, the concept of Augmented Reality is not a new one: Ivan Sutherland and Bob Sproull developed the first prototype AR system in 1968 [2]. This consisted of a see-through head mounted display, a mechanical tracking system, a computer and custom graphics hardware.

A typical Augmented Reality system consists of three subsystems:

1) A tracking mechanism to perceive,recognize and track the real world.
2) Graphics hardware to generate the virtual layer.
3) A display that combines the real world with the virtual layer.

Note that the display can range from a regular smartphone display to a see-through display on which only the virtual layer is shown. The tracking mechanism also differs between devices. In the ideal case, a 3D-camera is used, in addition to device sensors such as accelerometers and gyroscopes. That way, depth can be perceived and the locations at which virtual objects need to be placed can be accurately recognized and tracked. However, most consumer devices currently still have a conventional 2D camera, without another way of tracking depth. When no depth perception is present, there is no way to determine the correct size that the objects in the virtual layer should have. Using physical markers in the real world solves this problem. These markers provide a way to determine the location, orientation and translation of the virtual objects. As a result, they will be correctly scaled and positioned.

B. Building Information Modeling

Within the construction industry, the leading platform when it comes to 3D visualization and data storage for projects is Building Information Modeling (BIM). First implemented 30 years ago, it is now a universal standard used for construction projects of all scales around the world.

The essence of BIM is to combine 3D models with all types of auxiliary information about a project such as materials, costs and lifespan. [3] The result is a tool that can greatly impact the efficiency of the work flow for large projects, thus allowing for smoother development. Because BIM is such an invaluable tool, it is widely used. This presents an opportunity for the 3D visualization of renovations, as a lot of the required information already exists in BIM-compatible models. These models can be acquired from manufacturers, or from public libraries such as the one at BouwConnect Bibliotheek [4].

BIM is not a unified standard, as such there are several different industry standards which have their own file extensions and specifications. Common formats are DWG (Drawing), DXF (Drawing Exchange Format) [5], and IFC (Industry Foundation Classes) [6].

Both DXF and DWG are standards that were developed by AutoCAD, however DXF is an open source interchange format whereas DWG is a proprietary format. However DXF is an old standard, originally developed in the eighties (alongside AutoCAD 1.0) and therefore does not hold up anymore in modern day development. IFC is an open source standard developed through the cooperation of several industry leaders. Due to its nature as a modern open file format, there are a lot of existing open source solutions that deal with processing these files in some way.

For the purpose of our app we will be focusing on open source formats as proprietary formats do not grant us the flexibility and ease of use to allow us to develop with them in the limited timespan of the project. This limits us to DXF and IFC, where the former is an outdated format that does not accommodate for all the information that modern building projects require. Thus IFC seems to be the most applicable candidate for use with our app.

V. AR Platforms

In this section, we compare the existing mobile AR platforms to the requirements that have been defined in the previous section, in order to find the best platform for our goals.

A. ARToolkit

ARToolkit [7] is an open-source AR SDK that uses marker-based tracking. Using ARToolkit, it is possible to track multiple markers simultaneously and add 3D objects on top of them. OpenGL ES functions can directly be used, which means that developers can implement their own model loaders. A Wavefront .obj loader is also provided. ARToolkit can be used with Android, iOS and Windows phone (and desktop systems). Because of these features and the possibility to extend those features, ARToolkit is a suitable platform for our application.

B. Vuforia

Vuforia [8] is an AR SDK that supports Android and iOS. It provides both marker-based and markerless tracking. Like ARToolkit, OpenGL ES functions can be directly used and developers can implement their own model loaders. Because of that, it could be a suitable platform for our application. One downside of this platform is the one-time fee of $500.

C. Wikitude SDK

Wikitude [9] is another SDK for Android and iOS that provides both marker-based and markerless tracking. However, models must be in the proprietary .wt3 format, which can only be created by manually importing model files. Because our application should be able to import new model files (in other formats than .wt3) or download them from a database, this platform is not a feasible option.

D. Kudan SDK

Kudan SDK [10] is available for Android and iOS and provides both marker-based and markerless tracking. However, like the Wikitude SDK, it uses its own proprietary model format. As such, it is not suitable for use in our application.

E. Pikkart AR SDK

Pikkart AR SDK [11] offers marker-based tracking for Android and iOS devices. Because not enough documentation and support is available, this platform is not suitable for our application.

F. AR-media SDK

AR-media SDK [12] is available for Android and iOS and offers multiple tracking methods: 3D object tracking, planar tracking, location tracking and motion tracking. However, testing the demo application that was provided made this platform seem to function worse than others (there was more lag observed and the detection did not work quite often). Because of that, this platform is not a suitable choice for our application.

G. Augment

Augment [13] is an AR platform that aims to provide a full solution to manufacturers that want consumers to be able to visualize their products. 3D models can be uploaded to the Augment website, after which they are usable in the app. Due to the markerless tracking that is used, the size of the models does not relate to the size in real life. Because both automatically importing models and correct size are very important, this platform can not be used for our application.

H. BeyondAR

BeyondAR [14] uses the geographical location of the user to visualize 2D models. Because both markers and 3D models are not supported by this platform, it is not suitable for our application.

I. mixare

Mixare [15] uses (only) the geographical location of the user to determine the virtual objects. In our application we will use markers to visualize these objects, so this platform is not suitable for our purpose.

J. DroidAR

DroidAR [16] is an Android framework that makes it possible to use both location and marker based Augmented Reality. Because the development has stopped in 2013 and the documentation is lacking, this platform is not a feasible option for our application.

K. ARLab SDKs

ARLab [17] has released numerous SDKs, (among others) for image recognition, image tracking and 3D model placement. However, the image tracking SDK that would be required for the application is not yet available for Android. At the time of writing, most SDKs have been removed from the ARLab website, making them unaivalable as well. Because of that, these SDKs are not usable for our application.

VI. Plan of Approach

Our codebase will be hosted in a git repository. The main branch will always be stable, each feature requires its own feature branch for development that will only be merged into the main branch once it is fully functional. Before a feature can be merged, it needs to be reviewed by the rest of the team through the use of a pull request.

The IDE we will use for development is Android Studio. This is currently the officially supported development suite for Android.

Android software is based on Java, but we will also be using native code (C and C++) in order to work with our toolkits and APIs. For Java we will use the Official Android Code Style.

Due to the relatively short duration of this project, in addition to the uncertainty when it comes to the completion of product features, we are opting to use the Scrum method for our development. This gives us the flexibility that we need in order to complete this project successfully.

VII. Conclusion and Discussion

When comparing the requirements of the application and the available AR platforms, most platforms turn out to be unsuitable. The most common issues revolve around the lack of support for marker based tracking or the troublesome (manual) addition of models to the app. Another aspect that sometimes lacked was the availability of clear documentation or examples. Some platforms also seemed to be abandoned and no longer developed or supported.

In the end, two platforms that are suitable for our application remain: ARToolkit and Vuforia. Vuforia costs a one-time fee of $500 and offers more features than ARToolkit. However, since we do not need those features and the marker tracking of ARToolkit seems to have a sufficient performance, we have decided to use ARToolkit.

The only requirement that is not supported by ARToolkit is that the application must be able to visualize models from BIM formats. This will be solved by converting the IFC files to files that are supported by ARToolkit, or by writing a model loader so that the IFC files can be directly read by our application.

REFERENCES

[1] R. Azuma, "A survey of augmented reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997.

[2] M. Billinghurst, A. Clark, and G. Lee, "A survey of augmented reality," *Foundations and Trends in Human-Computer Interaction*, vol. 8, no. 2-3, pp. 73–272, 2014.

[3] E. Green, "Bim 101: What is building information modeling?" [Online]. Available: http://www.engineering.com/BIM/ArticleID/11436/BIM-101-What-is-Building-Information-Modeling.aspx

[4] [Online]. Available: http://bcb-online.nl/

[5] [Online]. Available: http://www.scan2cad.com/dxf/file-specification/

[6] [Online]. Available: http://www.buildingsmart-tech.org/specifications

[7] [Online]. Available: https://www.artoolkit.org

[8] [Online]. Available: https://www.vuforia.com

[9] [Online]. Available: https://www.wikitude.com

[10] [Online]. Available: https://www.kudan.eu

[11] [Online]. Available: http://www.pikkart.com

[12] [Online]. Available: http://dev.inglobetechnologies.com/features

[13] [Online]. Available: http://www.augment.com

[14] [Online]. Available: http://beyondar.com

[15] [Online]. Available: http://www.mixare.org

[16] [Online]. Available: https://bitstars.github.io/droidar

[17] [Online]. Available: http://www.arlab.com

B

Project Description

B.1. Project description

Private house owners who want to renovate their houses are increasingly looking for sustainable solutions. Possible options are insulation, use of sustainable materials and low energy applications for heating and ventilation.

Traditionally they involve contractor, installer and/or architect. However, making existing buildings more sustainable is a complex matter and not so many parties have experience. The offered solutions are therefore often disappointing for the owner. Our company Energiepaleis is specialized in designing, assisting and organizing sustainable renovation.

One of the issues we encounter is that the sustainable products are hard to visualize. Our clients wants to know how a heat pump fits into the building, what space does the ventilation system occupies and how solar panels can be arranged on the roof, etc.

To accommodate them, we show our clients existing renovated buildings or visiting centers so they can understand what may be the impact. This is, however, time consuming and not ideal, because it seldom corresponds fully with the situation and solution of the client.

In this project we are looking for an application (for example an app for smart phone or tablet) - based on Augmented Reality - that visualizes the required sustainable products in the client's house. With such a tool our clients will be better equipped to make informed choices. To our knowledge, such dedicated product is not yet available. Nevertheless, many manufacturers of sustainable products or materials already have a digital library with BIM product specifications. Ideally, the solution (application) does not need any or only little specific hardware and makes use of existing software (platforms), and it has a flexible design so that fast developments can be integrated easily.

Our request is to research how such application can be designed and constructed. We see the following steps as part of the project:

- What options to visualize exists that will offer the client a fair insight of their renovated house, and is flexible in its use?
- In what way can Augmented Reality play a role in such application, what are current existing similar uses of AR?
- Which platforms are (potentially) available to use AR for such applications?
- In what way can the building be visualized so that it supports adding an AR layer; what are its potentials and limitations?
- Design and construction of a first (simple) version of the application.

B.2. Company description

Energiepaleis is a start-up, founded in 2014 by Kees Stap and Hans de Wind. The company focuses on sustainable private home renovations. The company was founded with the notion that the market is developing slowly and acceleration is possible. In the vision of Energiepaleis there is room for a new role in the chain that takes responsibility for marketing and delivering integrated solutions. Energiepaleis will further develop this role and developed early 2015 a business model and plan. Energiepaleis provides practical services to private owners. We also develop products to support the necessary process innovation. You can find more information on: www.energiepaleis.nl.

C

Software Improvement Group Feedback

C.1. First SIG submission

What follows is the feedback we received from our first submission to the Software Improvement Group on June 12th:

”De code van het systeem scoort 4,5 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is.

De aanwezigheid van test-code is in ieder geval veelbelovend, het volume van de testcode blijft alleen nog wel sterk achter bij de hoeveelheid productiecode. Hopelijk zal het volume van de test-code ook groeien op het moment dat er nieuwe functionaliteit toegevoegd wordt.

Over het algemeen scoort de code dus bovengemiddeld, hopelijk lukt het om dit niveau vast te houden tijdens de rest van de ontwikkelfase.”

Thanks to this feedback we improved our test coverage.

D

Info Sheet

The following page is the info sheet for the project.

Title: AR for Sustainable Renovation of Buildings
Client: Energiepaleis
Final Presentation Date: 13 July 2017

Description

Description The task that was presented to us by the gentlemen of Energiepaleis was to create an augmented reality application designed to be used on a smartphone. This application needed to be able to show 3D representation of installations within a customer's home, with the emphasis on accurate size and shape.

In the research phase we learned that there are existing accurate models for the installations that need to be visualized. Additionally we researched various different platforms that we could build our application on top of. We settled on ARToolKit as the framework of choice, and the goal was to visualize existing models accurately in real-time, using marker recognition.

Development took place using a very lightweight variant of scrum in order to keep overhead to a minimum and maintain a stable version of the application. A big challenge we ran into was to properly utilize the existing models, which we overcame by converting them and making a compromise when it comes to the quality of the visualization. Additionally, converting models within our application proved to be extremely problematic due to outside dependencies.

Our final product is an Android application that allows the user to choose a 3D model from a selection of models, and link it to a real-world marker. The application then uses the phone's camera to visualize the model in the real world on the screen, overlaying the camera's image. We tested our application with a combination of manual testing for the UI components, and automated testing through AndroidTest and regular unit testing.

There is still a number of ways in which the application could be improved, such as the quality of the models, the addition of markerless tracking and the development of versions targeted at more specialized hardware. However, we've successfully managed to create a usable application that works on a wide set of regular smartphones.

Team Members

Name: Daniël van den Berg
Interests: Computer Graphics, Computer Engineering
Contributions: Back end developer, front end developer

Name: Jeffrey Helgers
Interests: Computer Graphics
Contributions: Head testing, front end developer

Name: Marcel Kralt
Interests: Computer Graphics, multimedia content creation
Contributions: Native developer, front end developer

Name: Sonny Lans
Interests: Math, Computer Graphics, complexity, algorithms, logic
Contributions: Native developer, front end developer, marker design

Project Affiliates

Client: Chris Hendriks
TU Coach: Elmar Eisemann Computer Graphics & Visualization Department
Contact Person: Sonny Lans sonny_lans@hotmail.com

The final report for this project can be found at: http://repository.tudelft.nl

Bibliography

[1] . URL https://developer.android.com/guide/components/activities/intro-activities.html.

[2] . URL https://developer.android.com/topic/libraries/support-library/index.html.

[3] . URL https://developer.android.com/about/dashboards/index.html.

[4] URL https://archive.artoolkit.org/documentation/doku.php?id=4_Android:android_developing.

[5] URL http://www.boost.org/.

[6] URL http://www.energiepaleis.nl/.

[7] URL https://google.github.io/android-testing-support-library/docs/espresso/index.html.

[8] URL https://github.com/.

[9] URL http://www.augment.com/blog/infographic-lengthy-history-augmented-reality/.

[10] . URL http://ifcopenshell.org/.

[11] . URL http://www.ifctoolsproject.com/.

[12] URL http://site.mockito.org/.

[13] URL https://www.nuclino.com/.

[14] URL https://www.opencascade.com/.

[15] URL http://robolectric.org/.

[16] URL http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf.

[17] URL https://developer.android.com/ndk/guides/standalone_toolchain.html.

[18] D. W. F. van Krevelen and R. Poelman. A Survey of Augmented Reality Technologies, Applications and Limitations. *The International Journal of Virtual Reality*, 9(2):1–20, June 2010.