Technische Universität Berlin

Faculty VI Planning Building Environment

Institute of Geodesy and Geoinformation Science

Fakultät VI
Straße des 17. Juni 52
10623 Berlin

Master Thesis

Integration of CityGML-based building data into an AR environment on a street view level

Salih Yalcin

Matriculation Number: 409406

02.04.2023

Supervised by
Prof. Dr.-Ing. Martin Kada
Prof. Dr.-Ing. Marc Oliver Löwner

First of all, I would like to thank Prof. Dr.-Ing. Martin Kada, head of the Chair of Methods of Geoinformation Science for giving me the opportunity to carry out state-of-the-art research in this field, and Prof. Dr.-Ing. Marc Oliver Löwner for his support and feedback in his role as my second supervisor.

Special thanks to my friends in TU Berlin. Also a big thanks to my colleagues at KAYAK. They allowed me to gain industry feedback and work on my thesis in parallel to my daily tasks and responsibilities at KAYAK.

Last but not least, thanks to my wife Fatma SENER YALCIN and our little boy Erdem YALCIN for their mental help and for being with me on this journey. They always made me smile, they gave me encouragement every time whenever I lost my motivation. Additionally, I greatly appreciate that this master's study and thesis were carried out with the financial support and contribution of the Republic of Türkiye.

I hereby declare that the thesis submitted is my own, unaided work, completed without any unpermitted external help. Only the sources and resources listed were used.

Berlin, 02.04.2023

. . . .
(Signature Salih Yalcin)

Abstract

In recent years, the representation of real-world environments has gained greater significance. These representations, which are three-dimensional models of urban environments and their structures, objects, and components, are known as 3D City Models. They are used for visualization, indoor navigation, infrastructure planning, 3D cadastral mapping, and urban planning. 3D City Models can be derived from a variety of techniques, such as photogrammetry, geographic information systems, synthetic aperture radar, and volunteered geoinformation.

Despite the availability of various data formats for storing, exchanging, and visualizing 3D City Models, CityGML and CityJSON are open data formats from the Open Geospatial Consortium that aim to facilitate the integration of urban geographical data for various applications. CityGML defines a conceptual model, while CityJSON is intended to provide greater flexibility.

Augmented Reality (AR) is a technology that overlays digital information on the real world using a smartphone's camera and other capabilities. While many AR applications have been developed for various purposes, AR and mobile applications may have additional potential in other fields. For example, in urban planning, AR could be used to enhance decision-making efficiency.

In this thesis, a data integration approach was implemented for 3D City and urban models. The integration of CityGML-based building data into an AR environment on a street view level was achieved through a client app that was developed using AR libraries from companies such as Google.

The results indicated that the integration of CityGML building data into an AR environment is an optimal solution, and the efficiency of this approach could be improved by using high-level posing algorithms in mobile apps.

Zusammenfassung

In den letzten Jahren hat die Darstellung von realen Umgebungen an Bedeutung gewonnen. Diese Darstellungen, bei denen es sich um dreidimensionale Modelle von städtischen Umgebungen und ihren Strukturen, Objekten und Komponenten handelt, werden als 3D-Stadtmodelle bezeichnet. Sie werden für die Visualisierung, die Navigation in Innenräumen, die Planung von Infrastrukturen, die 3D-Katasterkartierung und die Stadtplanung verwendet. 3D-Stadtmodelle können aus einer Vielzahl von Techniken abgeleitet werden, z. B. aus der Photogrammetrie, aus geografischen Informationssystemen, aus dem Radar mit synthetischer Apertur und aus freiwilligen Geoinformationen.

Trotz der Verfügbarkeit verschiedener Datenformate für die Speicherung, den Austausch und die Visualisierung von 3D-Stadtmodellen sind CityGML und CityJSON offene Datenformate des Open Geospatial Consortium, die die Integration städtischer Geodaten für verschiedene Anwendungen erleichtern sollen. CityGML definiert ein konzeptionelles Modell, während CityJSON eine größere Flexibilität bieten soll.

Augmented Reality (AR) ist eine Technologie, bei der digitale Informationen mithilfe der Kamera eines Smartphones und anderer Funktionen über die reale Welt gelegt werden. Während viele AR-Anwendungen für verschiedene Zwecke entwickelt wurden, können AR- und mobile Anwendungen in anderen Bereichen zusätzliches Potenzial haben. So könnte AR beispielsweise in der Stadtplanung eingesetzt werden, um die Effizienz der Entscheidungsfindung zu verbessern.

In dieser Arbeit wurde ein Datenintegrationsansatz für 3D-Stadt- und Stadtmodelle umgesetzt. Die Integration von CityGML-basierten Gebäudedaten in eine AR-Umgebung auf Straßenansichtsebene wurde durch eine Client-App erreicht, die unter Verwendung von AR-Bibliotheken von Unternehmen wie Google entwickelt wurde.

Die Ergebnisse zeigten, dass die Integration von CityGML-Gebäudedaten in eine AR-Umgebung eine optimale Lösung ist und dass die Effizienz dieses Ansatzes durch die Verwendung von High-Level-Posing-Algorithmen in mobilen Apps verbessert werden könnte.

Contents

List of Figures

List of Tables

Abbreviations

3D	Three Dimensional
AAB	Android App Bundle
AR	Augmented Reality
API	Application Programming Interface
APK	Android Package
BIM	Building Information Model
CityGML	City Geography Markup Language
CityJSON	City JavaScript Object Notation
COLLADA	Collaborative Design Activity
GML	Geography Markup Language
HMD	Head Mounted Display
IDE	Integrated Development Environment
JSON	Java Simple Object Notation
LIDAR	Light Detection and Ranging
LoD	Level of Detail
MR	Mixed Reality
OBJ	Wavefront Object
OFF	Object File Format
OGC	Open Geospatial Consortium
OHA	Open Handset Alliance
PLOY	Polygon Data Files
SAR	Syntetic Aperture Radar
SDI	Spatial Data Infrastructure
SDK	Software Development Kit
SLAM	Simultaneous Localization and Mapping
UI	User Interface
UX	User Experience

VPS	Virtual Positioning System
VR	Virtual Reality
XML	Extensible Markup Language
XR	Extended Reality

1 Introduction

1.1 Motivation

In recent years, the use of mobile apps has exploded in popularity, with billions of people around the world relying on them for a wide range of purposes such as communication, entertainment, and accessing information. This trend is expected to continue in the future, making mobile apps a vital platform for delivering content and services to users.

3D city models, on the other hand, are digital representations of urban environments that capture the geometric, topological, and semantic aspects of real-world cities. These models have a variety of applications, including urban planning, disaster management, and GIS-based services. By using 3D city models, professionals and decision-makers can benefit from a more accurate and realistic portrayal of cities, leading to improved outcomes.

Augmented reality (AR) is a technology that superimposes digital information onto the physical world in real time. When combined with 3D city models, AR can provide an immersive and interactive experience for users, allowing them to visualize and explore the city in a new way. CityGML and CityJSON are open standards developed by the Open Geospatial Consortium specifically for 3D city models, and they facilitate the exchange and integration of 3D city model data between different systems and platforms.

The combination of mobile apps, 3D city models, and augmented reality offers numerous opportunities and applications. The adoption of open standards like CityGML and CityJSON is crucial for ensuring the interoperability and compatibility of 3D city model data, which is essential for fully realizing the potential of this technology. This makes the topic of 3D city models and augmented reality with city information a timely and relevant subject for a master's thesis. The potential benefits of this technology are vast and varied, and further research in this area has the potential to lead to significant advances in a range of industries.

1.2 Goal

This master thesis aims to explore an answer to the integration of 3D City and planning models into urban AR applications. For this purpose in mind, a data integration approach is needed. Although many applications have been developed for AR, the focus will be the integration of these data for urban applications.

The goal is to develop an Android app, that is having CityGML-based building data integrated into an AR environment on a street view level.

The challenging part of this development process is the integration of the 3D City and planning models that need to be performed on the either client side or the server side. Also, data loss is the most crucial thing that should have to be taken care of when converting CityGML to OBJ.

1.3 Methodology

In order to find an answer to the research questions and formulate those answers, the methodology has been split up into three components. In the beginning, a literature study will be done on 3D City Models, AR, and 3D data formats for the representation of 3D City Models. In addition to those points, the current state of the art and the developments will be described. Questions about the use, requirements, and different interests of 3D city models will also be examined. The second component of the methodology explains why and which data format was selected as well as the development platform selection. The implementation phase aims to elaborate the development of a mobile app that integrates CityGML-based building data into an AR environment on a street view level.

2 Related Work

This Chapter will enhance the understanding of the subject matter, an assessment will be made of the studies related to the thesis topic. In addition to providing a general overview of 3D city models, an attempt will be made to uncover how presentations in various formats have been rendered in regard to the visual representation of these models and their augmented reality applications. Initially, the discussion of 3D City models will be addressed in Section 2.1, and the usage of these models in Augmented Reality applications will be explained in Section 2.2. Finally, given that the CityGML and CityJSON were used in the developed application for the purposes of the thesis, Section 2.3 will investigate how those data formats were working on behalf of using them in the augmented reality world.

2.1 3D City Model

A 3D City Model is a geometric representation that allows the real world to be expressed in a digital form. A common 3D City Model could be created from several techniques, for instance, extrusion from 2D footprints, architecture, photogrammetry, and volunteered geoinformation. [1] In order to get an accurate 3D City Model there are some techniques that provide that information. Some of them are photogrammetry, LIDAR, and SAR which get those data without contacting the real world. [2]

However, technology is developing year by year, and 3D city models are gaining importance in several research areas. Those areas' purpose is to give more valuable information than visualization and it can be categorized among a variety of applications. As shown in Figure 1, the utilization could be summarized as follow; from infrastructure planning to 3D cadastre, utility management to solar potential estimation could be some of those application areas. [3]

Although 3D city models play a crucial role in the current projects there will be more application use cases where people have benefited from the combination of real-world and augmented world. There will be more demand for these kinds of applications as shown in Figure 1. [4]

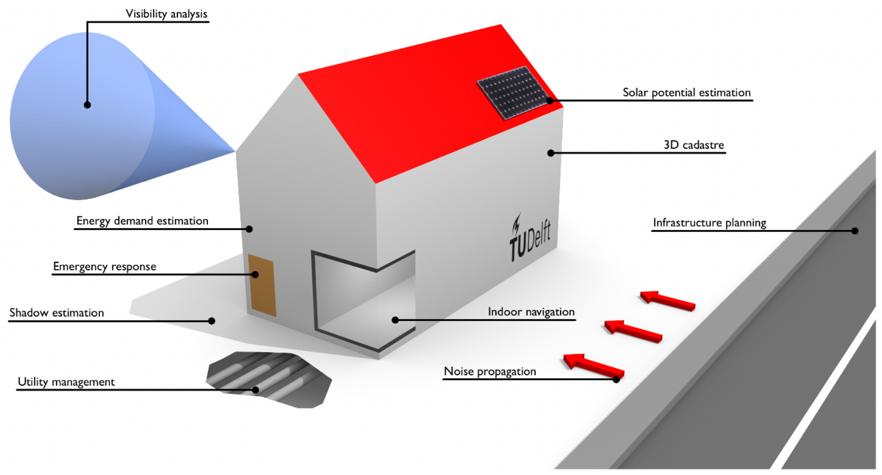


Figure 1: 3D City model applications
[1]

2.2 Augmented Reality

AR is a technology that combines the real world with the virtual world. This technology aims to enrich the real-world experience by exploring integration within a digital environment. The more developments in the computer vision area the more valuable AR to be a more in the future. [5, 6]

Although AR is defined as a virtual technique that combines virtual content with the real world, which is recognizable mostly with sight, it's also be applied to the other senses such as smelling, touch, and hearing. [7] AR achieved popularity with the help of developments in technology like depth cameras and computer vision. There are other popular reality technologies that combine virtual and physical worlds for various use cases. These include Virtual Reality (VR), Mixed Reality (MR), and Extended Reality (XR). VR uses the virtual world to create a separate world from the real world and the user has a chance to interact with that world by not contacting the real world. VR is also suitable in indoor environments to interact with. [8] MR is a technology that mixes virtual content with the real world but also allows interaction between them. In the MR experience, the user can see and interact with both the digital elements and the physical ones. Therefore, MR experiences get input from the environment and will change according to it. [9] Finally, XR is an emerging technology that covers VR, AR, and MR technologies and is referred to as an umbrella that brings all the reality technologies under one term as shown in the 5. [10]

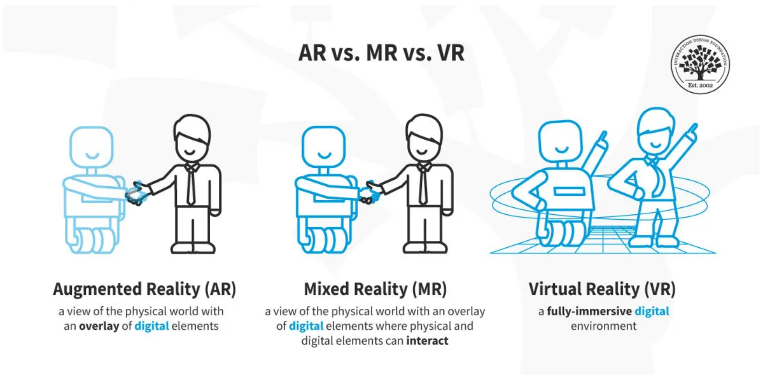

AR vs. MR vs. VR
Est. 2002
Augmented Reality (AR)
a view of the physical world with an overlay of digital elements
Mixed Reality (MR)
a view of the physical world with an overlay of digital elements where physical and digital elements can interact
Virtual Reality (VR)
a fully-immersive digital environment

Figure 2: The differences between reality technologies
[11]

2.2.1 Use Cases and Challenges

There is a wide range of use cases and challenges in AR applications. In this section, the information is briefly discussed with the help of the literature.

Predescu et al. developed a platform for gaming purposes using geospatial aspects. They found that the achievement of the implementation phase was difficult due to there being many frameworks, commercial ones and some of them were experimental. All in all, they also found that real-world testing is challenging in order to do some calibration. [12]

Portales et al. initiated a tentative and low-cost outdoor mobile AR application. The 3D photo models were derived from photogrammetry and they have been integrated into the real world. The introduced prototype includes required a head-mounted display (HMD) and user navigation has been achieved by sensors. The study revealed that their application could be useful for several purposes. Also, they have mentioned that using those applications would be cost efficient than stand-alone virtual modeling. [13]

Sanaeipoor et al. proposed an AR strategy to place marking and its effects on urban projects. The overall solutions seem to expose that this technology is a useful tool in the field of particular place marking and urban design processes. The paper also demonstrates that AR technology could cover all the people in charge of public arts as well as the involvement of citizens. [14]

Latino et al. developed a prototype for an indoor augmented reality app that aims to contribute to stakeholders' decisions which supports them to pitch urban design ideas. Their app allows them to visualize 3D objects on flat surfaces. Developed app tested among test group which is established by the development team yet to emphasize that this point needs to be improved. [15]

Another augmented app was developed by Cirulis et al. in order to supply architecture and urban planning. In order to visualize 3D op Their research revealed that 3D data is crucial for augmented reality apps. 3D AR applications allowing to understand a city with 3D buildings. [16]

Muthalif et al. have made a comprehensive study to investigate the state art of subsurface utilities. The visualization techniques have been analyzed in four parts. XRay view, transparent view, shadow view, and topo view. All parts of this study have been compared in terms of quality of depth perception, occlusion of the real world, visualization complexity, and parallax effect. Their study has shown that no single method provides the best visualization in all application scenarios depending on the type of the application and user requirements. [17]

Positioning on AR apps could be challenging because it requires estimation and updating real-time of dynamic augmented scenes in. Wang et al. have proposed a quaternion-based visualization approach to the 2D and 3D vector data. The study showed that position could be possible with a quaternion-based approach. Also, in order to improve vector visualization accuracy, the proposed method fits the AR applications. [18]

Creating virtual environments based on geospatial data is also up to date subject. Keil et al. has created a virtual environment from open geospatial data. Their work summarizes that web technologies are having support worldwide communication and the exchange of data. The rise of spatial data available for developers makes virtual applications more promising. However, some geospatial data formats are not supported by game engines. That's why they have provided how public geospatial data can be transferred to file formats that can be understandable by visualization engines. [19]

2.2.2 General Applications

Augmented Reality is a technology in which virtual objects are projected into the real environment and many data such as location information, 2D/3D data, video, and audio are transferred. This technology is capable of meeting both the social and individual needs of users, supporting their education and professional development, as it contains advanced software and hardware elements used today with its wide range of effects. [20]

AR is playing a significant role in many fields such as logistics, education, architecture, sports, and education.

The use of augmented reality in logistics is mostly found in procurement, storage, packaging, and transportation processes. The use of augmented reality technology in the logistics sector has brought a different perspective to this field. There are many advantages as well as disadvantages in the logistics sector where augmented reality applications are used. While the use of augmented reality in the logistics sector has advantages such as saving time, reducing waste, and determining the fastest route; there are also disadvantages such as the inability to ensure the confidentiality of the employees in the sector, the negative impact on the health of the employees in long-term use of the application tools, and the high cost of the tools and equipment used in the application. [21]

The use of augmented reality technology in architecture; 3D visualization of design models, quality control, and inspections using different layers of information, supporting a democratic and collaborative approach supports more successful project formation. Developed companies use augmented reality technology in architecture, especially to save time and budget. [22]

Educational applications are one of the main areas where augmented reality technology is used. These applications can be used as educational materials at all educational levels from primary education to university. Augmented reality technology is an environment where students can control their own learning by interacting with the virtual and real world. This technology has advantages for both educators and students. The use of augmented reality applications in the classroom increases student motivation and student contribution to learning outcomes. [23]

Augmented reality technology is used in many different branches by utilizing object tracking and identification technology in the field of sports. The purpose of the use is generally; to improve athletes' abilities, improve their performance, facilitate their training, ensure that they spend quality time, and increase their awareness of the games. Such as tennis, cricket, volleyball, soccer, and basketball. [24]

2.2.3 Geospatial Applications

Many industries experience the need for managing their geospatial data during the whole life cycle of their assets; especially since their business relies on geospatial data. This dependence makes AR technology in the geospatial industry compelling. In this section, several GeoAR applications will be explained.

Choi et al. described a way to organize and group geospatial data according to their tags. They have used manual and automatic approaches as well as the nearest neighbor algorithm. The study was carried out by implementing an AR application on an Apple platform. Their study showed that using AR has advantages, considering a precise but manual approach. [25]

Bharath et al. designed and developed an application to visualize geographical data through an augmented 3D model. Their work proposes a way to visualize geospatial globe data using AR technology. They have developed an android application the scan given marker to show the globe. [26]

Schall et al. developed mobile augmented reality for the visualization of underground infrastructure using geospatial data. Their study showed that the potential for improvements such as on-site planning, data capture, and survey planning is possible using AR with geospatial capabilities. They also stated that achieving the functionality of AR depends on localization, and integration of visual tracking in orientation sensor information. [27]

Badard has explained the concept of Geospatial Service Oriented Architecture and its relevance in the definition of distributed and interoperable multi-representation data infrastructures to enhance the experience of the users in mobile augmented reality applications. [28]

St-Aubin et al. developed a 3D collaborative geospatial augmented reality system for urban design and planning purposes, their contribution presents a system to resolve urban planning problems. The implemented framework allows for developers to simply and quickly create a 3D geospatial augmented reality system applied to desired urban planning scenario. [29]

Another AR application is annotating the geospatial model which is providing field workers with redlining capabilities which enable the outdoor user to annotate and interact with geospatial objects. The fieldworker can choose a symbol from a predefined palette of symbols. Besides placing an annotation to a location in the geospatial 3D model, the field worker also has the possibility to survey locations by intersecting poses with the model. Those developments help users to have a chance to validate the model interactively. [30]

Using geospatial approaches could also be applicable in the advertising area. Jamali et al. implemented an augmented approach for indoor advertising. Their study proposes a framework to apply the capabilities of Ubiquitous Geographic Information Systems and spatiotemporal modeling to determine the best location for installing advertising billboards and signs in indoor environments. They have used location-based services and have benefited from AR providing additional information about the ads and creating interactive content. [31]

Kasperi et al. propose and evaluates an alternative model-based method for dynamic outdoor AR of virtual buildings rendered on non-depth sensing smartphones. They have used geospatial data to construct the geometric model of real buildings surrounding the virtual building method and remove the target regions from the virtual building using masks constructed from real buildings. Their study also revealed that participants expressed that their experience improved when using their model. The solutions showed that using geospatial data for simulating occlusions is a sufficiently effective solution until depth-sensing AR devices are more widely available. [32]

2.2.4 Visualization Techniques

Application of AR for visualizing 3D buildings has led to many studies over the past two decades, Yet there are still significant improvements have to be made before AR can practically be used in 3D building visualization works. Different techniques introduced to enhance AR visualization of 3D buildings so far, can be classified into six main categories. Marker based visualization, X-Ray view, Transparent view, Topo view, Image rendering, and Cross section view. In the following, the discussion about the principles of the above AR visualization techniques will be made.

Marker based visualization, is a widely used technique for superimposing virtual 3D objects onto the real world. In this technique, a physical marker is used as a reference point to anchor the virtual object in the real world. The marker is usually a pattern or an image with high contrast and distinctive features that can be easily recognized by a camera. The camera captures the marker and computes its position and orientation, which is used to overlay the virtual object in the correct location in the real world.

One notable application of marker-based AR in architecture is for visualizing 3D buildings. For instance, Camba et al. developed a system and combined combined with traditional printed materials to enhance the visualization and understanding of technical information. Their method aims to create custom marker-based AR content using 3D data from real objects and authoring tool that they developed in house. The system allows instructors to quickly and effortlessly create their own AR content to support their innovative teaching practices. [33]

Another example is the use of marker-based AR for visualizing building models in the context of urban planning. Anagnostou et al. developed a system called SqquareAR, an Augmented Reality authoring application which enables virtual restoration of public, unexploited, spaces in cities. The system allows local residents participate into the decision making process that affects their everyday life. [34]

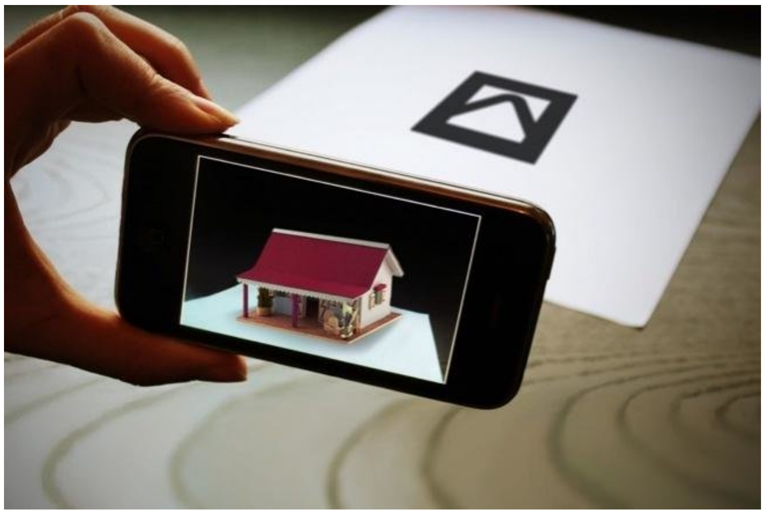


Figure 3: Marker based visualization
[35]

X-Ray visualization, is the most popular technique that has been examined in the past studies to visualize hidden objects/utilities. It provides visualization of a hidden object inside another visible object. X-Ray View can also be used for 3D building visualization, which allows architects, engineers, and urban planners to visualize the internal structure of buildings in a more intuitive way.

One of the ways X-Ray View is used for 3D building visualization is by integrating point cloud data obtained from laser scanning technology. This data can be used to create a 3D model of the building's interior, which can then be viewed and manipulated in X-Ray View. This allows users to see the building's internal structure and identify any issues or areas that need attention. Another way X-Ray View is used for 3D building visualization is by incorporating Building Information Modeling (BIM) data. BIM is a digital representation of the physical and functional characteristics of a building, and it can be used to create a detailed 3D model of the building's internal structure. By integrating BIM data into X-Ray View, users can explore and visualize the building's internal structure, as well as its mechanical, electrical, and plumbing systems. [36]

Research studies have shown that X-Ray View can be an effective tool for 3D building visualization. Avery et al. developed an extension x-ray vision system for augmented reality that employs multiple view models to support new visualizations to provide depth cues and spatial awareness to users. Their systems make hidden objects appear to be behind walls and gave user to understand real world in a more spatial way. [37]

Transparent View, is a visualization method that allows users to see through the external walls of a building to view its internal structure and layout. This is accomplished by creating a transparent virtual overlay that is superimposed onto the user's real-world view using augmented reality (AR) technology. It provides users with a better understanding of the building's internal layout and structure. This technique can be particularly useful during the design and construction phases, as it can help to identify potential issues before they become more costly to fix.

There have been several research studies that have explored the use of Transparent View for 3D building data visualization in AR. For example, a study by Gomez et al. examined the use of Transparent View to visualize transparent objects from multiple images with the aim of both discovering such objects and building a 3D reconstruction to support convincing augmentations. Their study shown that participants were able to use Transparent View to gain a better understanding of the internal structure of the building and its historical significance. [38]

Image rendering, is a visualization method that uses 3D models to create realistic images of buildings that can be integrated into the user's real-world view using AR technology. This approach can help to provide users with a more accurate and detailed view of the building's exterior appearance and features, such as its texture, color, and materials.

Wen et al. developed an AR system that renders images when using unmanned aerial vehicles to visualize 3D buildings as shown in Figure 4. Their study found the system has the potential for discovering unnoticed detail problems in real construction site, such as vehicle path planning conflicts and structure spatial dilemmas by image rendering on AR. [39]

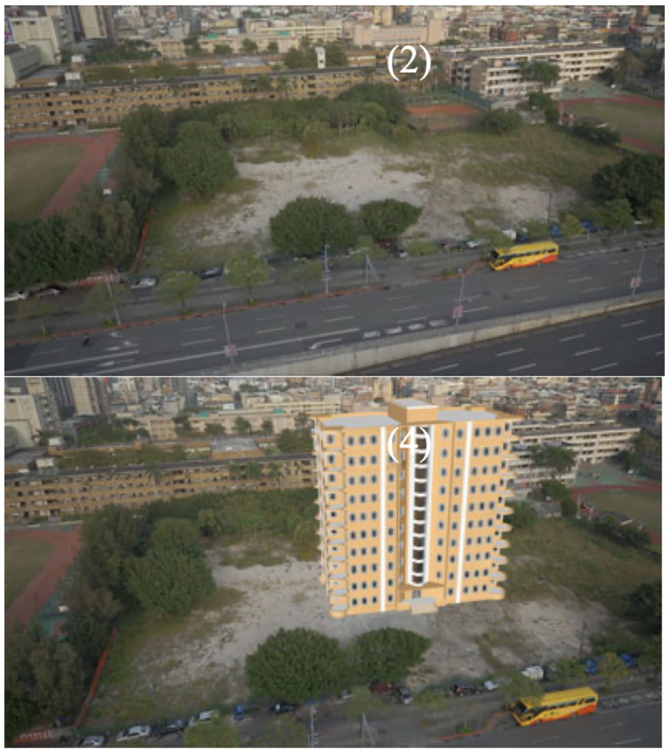


Figure 4: Image rendering visualization in AR
[39]

3 Background

In general, an Android application can be split into four essential components, activity, services, broadcast receivers content providers. The following section elaborates on fundamental methods for this thesis and related topics of mobile application development, especially with Android. Related techniques or terminology are additionally provided if required.

3.1 CityGML and CityJSON

CityGML is an OGC standard and defines a conceptual model and exchange format for storing, exchanging, and representing digital 3D City models. [Open Geospatial Consortium, 2012]. The aim of CityGML is to integrate geodata for a wide range of applications for smart cities and digital twins, as well as urban and rural planning. The CityGML standard has versions 1.0, 2.0, and the latest version 3.0 which is the evolution of the previous versions. Although previous versions standardized a GML exchange format while CityGML 3.0 standardizes the information model, and can be implemented in a variety of areas. [40]

The CityGML is modular, with various thematic modules that can be combined to create a profile that suits specific applications. The core module defines base classes for all features in CityGML, while the relief module provides terrain representations. The transportation module supports route planning and trajectory planning for mobile robots, and the water body module provides boundary surfaces for water bodies. The vegetation module models solitary vegetation objects and supports analysis tasks, while the land use module provides an areal representation of land use. The City Furniture module represents immovable objects like street lights, traffic lights, and flower containers, and the Generics module allows for the extension of the CityGML model. All features defined in the modules have attributes and geometrical representations in different Levels of Detail (LoDs), with the shared attribute relativeToTerrain explicitly representing the relation of a feature to the terrain surface, which is crucial for many applications. [41]

The CityGML data model could be decomposed into a core module and submodules such as bridge, building, CityFurniture, CityObjectGroup, Generics,

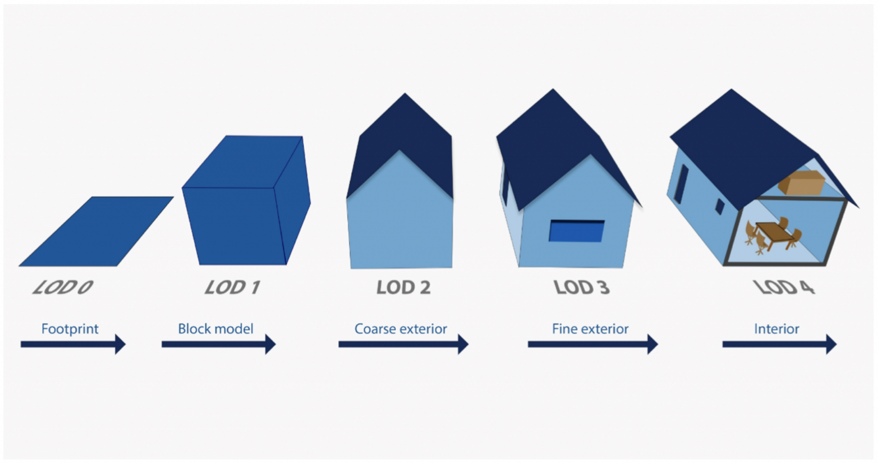


Figure 5: CityGML level of details

LandUse, Relief, Transportation, Tunnel, Vegetation, and WaterBody. Each sub-modules covers a different part of city objects. [40]

CityGML 2.0 has been adopted for a wide range of applications, and although it's providing powerful information, OGC brought a new CityGML version that combines CityGML 2.0 modules with newly added modules and as well as revised modules. New modules could be called on top of CityGML 2.0 modules with Dynamizer, Versioning, PointCloud, and Construction. Core, Generics, Building, and Transportation modules have been also revised. New CityGML 3.0 applies a model-driven approach in the creation of the data model and exchange formats. The new concept for CityGML counts all city objects to the space and space boundaries. Space could be identified as the real world's data such as buildings, water bodies, and trees. A Space Boundary is an entity with an areal extent in the real world. [42]

As shown in Figure 5, the CityGML also has 5 levels of details. LoD0 is to show footprints, LoD1 is to show a block model of the building. LoD2 to show the coarse exterior of the building. LoD3 is created to give information about the fine exterior of the building. In that part of the level of detail, the building's properties such as windows, and roofs are visible. LoD4 is showing the interior of the building on top of information coming from the other level of details. [40]

CityGML has three exchange format so far. The latest version is CityGML 3.0 is an improved version of 1.0 and 2.0. While older versions standardized as GML format, CityGML 3.0 standardized to be information model. Allows data to be stored in GML'X́ML format as well as JSON format. [43]

CityJSON is an open standardized data model for 3D Cities storing, exchanging, and representing, which is extended from CityGML(version 3.0) based on JSON. Although GML encoding CityGML offers complex encoding, the aim of

CityJSON is to offer an alternative way to be easy to use, faster, and compatible to platforms like the web and mobile. [44]

The first level of the CityJSON file contains the entire CityJSON object. The property metadata usually contains a reference system that always applies to the entire CityJSON file as described in Table 1 The property city objects contain a collection with key-value pairs. The key is the ID of the city object and the value is the city object. The property attributes handle normal attributes and the generic attributes. The property geometry of the city object contains an array of geometric objects. The property semantics of the geometric object contains a semantics object. The property surfaces of the semantics object contain an array of semantic surface objects. [45]

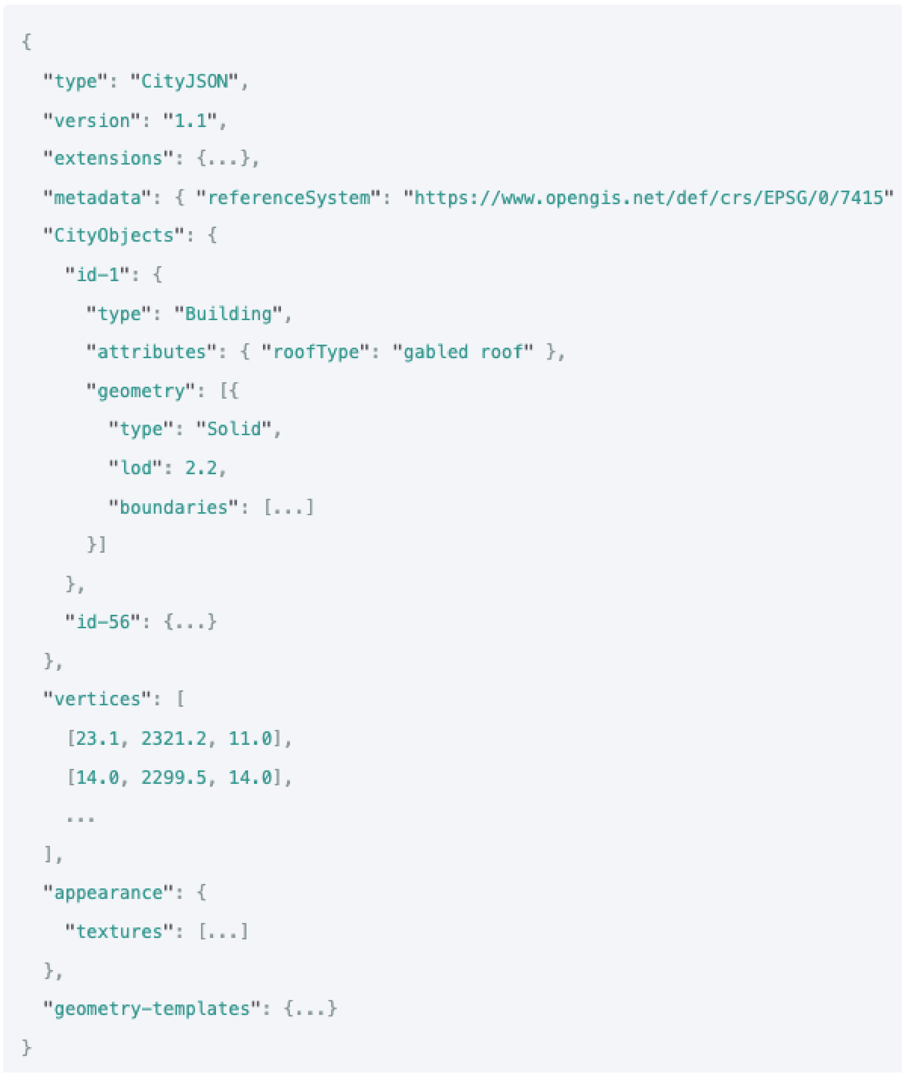
CityJSON Object Properties		
Name	Requirement	Description
type	must	The value must be string. "CityJSON"
version	must	The value must be string with the version (X.Y) of the used CityJSON schema.
metadata	may	The value is an object with properties describing the metadata.
cityobjects	must	The value contains a collection of key-value pairs. The key is the ID of the object and the value is city-object.
vertices	must	The value is an array with the coordinates of each vertex used in the CityJSON File
transform	may	The value is an object describing in which way the coordinates can be transformed to integer values.

Table 1: CityJSON file components

The structure of CityJSON is easily understandable as it could be read by humans shown in Figure 6

Upon reviewing the literature, it becomes evident that investigations and analyses utilizing CityGML and CityJSON have been conducted across numerous disciplinary applications, outputs various outcomes. One of them is Biljecki et al. have investigated whether an automatic conversion is possible among CityGML and OBJ files. Their investigation revealed that for simple conversions proposed method is compelling. However, the conversion method can not differentiate between a roof, and a terrace or garage top. The more requirements become advanced, the more complex method will require to do this automatic conversion. [46]

On the other hand, Blut has developed a mobile app that converts CityGML using XMLPullParser and shows used data on smartphones using a viewer that extended from game engine alternatives. The study emphasizes that showing CityGML on smartphones challenges handling complex 3D objects, not only in terms of memory but also storage space, mobile processing units, and display sizes. [47]



```
{
  "type": "CityJSON",
  "version": "1.1",
  "extensions": {...},
  "metadata": { "referenceSystem": "https://www.opengis.net/def/crs/EPSG/0/7415"
  "CityObjects": {
    "id-1": {
      "type": "Building",
      "attributes": { "roofType": "gabled roof" },
      "geometry": [{
        "type": "Solid",
        "lod": 2.2,
        "boundaries": [...]
      }]
    },
    "id-56": {...}
  },
  "vertices": [
    [23.1, 2321.2, 11.0],
    [14.0, 2299.5, 14.0],
    ...
  ],
  "appearance": {
    "textures": [...]
  },
  "geometry-templates": {...}
}
```

Figure 6: CityJSON file structure
[44]

Kolbe et al. has presented the method for storing and accessing 3D City Models using CityGML, especially in SDIs. Utilization of CityGML model concepts has been done and the research express that, CityGML provides substantial information. [48] Geo-referenced models look promising when it comes to creating augmented reality applications as well as mixed reality applications. Praschl et al. have created occlusion models for mixed-reality applications using Microsoft HoloLens. Their study indicated that geometry file formats such as OBJ or COLLADA are used commonly but they lack semantic information. Their implementation also showed that the CityJSON format for exchanging spatial information looks promising. [49]

Last but not least, visualization for OGC standardized data looks promising when it comes to web-based visualization. Mohad Hanafi et al. has expanded 3D object depiction by implementing CityJSON-based encoding. The main focus was showing 3D spatial data, especially building and parcel using a web platform and their enhancement revealed that visualization with CityJSON has succeeded with display, query, and update attributes. [50]

3.1.1 Software and Libraries

A variety of tools and software have been utilized to analyze, visualize, compare, convert, manipulate, and validate the CityGML and CityJSON formats. In this section, an overview of both open-source and commercial software as well as libraries are given.

- **citygml4j** is an open-source Java library and API for OGC CityGML. It was created with the intention to make CityGML processes like parsing, editing, and writing easier. It supports CityGML 3.0 Conceptual model and CityGML AEDs [https://github.com/citygml4j/citygml4j]

- **citygml-tools** is a command line tool that bundles several operations for processing CityGML files which include statistics, validation, filtering lods, upgrading, and conversion [https://github.com/citygml4j/citygml-tools]

- **cjval** is a library to validate CityJSON objects. It's supporting the validation of CityJSON and CityJSON features. [https://github.com/cityjson/cjval]

- **ninja** is a web viewer for CityJSON files. It supports CityJSON file specifications. [https://github.com/cityjson/ninja]

- **azul** is a 3D city model viewer that was developed specifically for apple silicon macs. The intention behind this app is to view CityGML files from version 1.0 to 2.0 and CityJSON 1.0 to 1.1. Also, it supports viewing OBJ, OFF, IndoorGML, and POLY files. It supports loading multiple files, selecting objects by clicking them or selecting them in the sidebar, toggling the visibility of individual items, and browsing their attributes.[https://github.com/tudelft3d/azul]

- **cityjson-qgis-plugin** is a Python plugin for QGIS 3 which adds support for loading CityJSON datasets in QGIS. [https://github.com/cityjson/cityjson-qgis-plugin]
- **cjio** is a Python command line interface program. It supports CityJSON operations like manipulation, translation of coordinates, removing attributions, texture processes etc. [https://github.com/cityjson/cjio]
- **FME Workbench** is a software tool that has been used to manipulate and transform geospatial data. Its ability to integrate with different data sources, comprehensive features, and user-friendly interface make it a valuable tool for managing and transforming geospatial data. [https://www.safe.com/fme/fme-desktop/]
- **QGIS** is a free and open-source geographical information system to deal with geospatial data. It can create, edit, visualize, analyze, and publish geospatial information on all platforms. [https://qgis.org/en/site/]
- **Cesium JS** is an open-source JavaScript library for creating 3D globes and maps with the best possible performance, precision, visual quality, and ease of use. It supports CityGML processes. [https://cesium.com/platform/cesiumjs/]

3.2 Android Development

Android is a free, open-source operation system specially developed for mobile devices. [51] The Android Operating system introduced in 2007 by Google joined forces with the mobile industry to form the Open Handset Alliance (OHA). Their mission was to establish Android as an open-source OS. This means anybody could access the sources and have a chance to modify the operating system to fit requirements. [52]

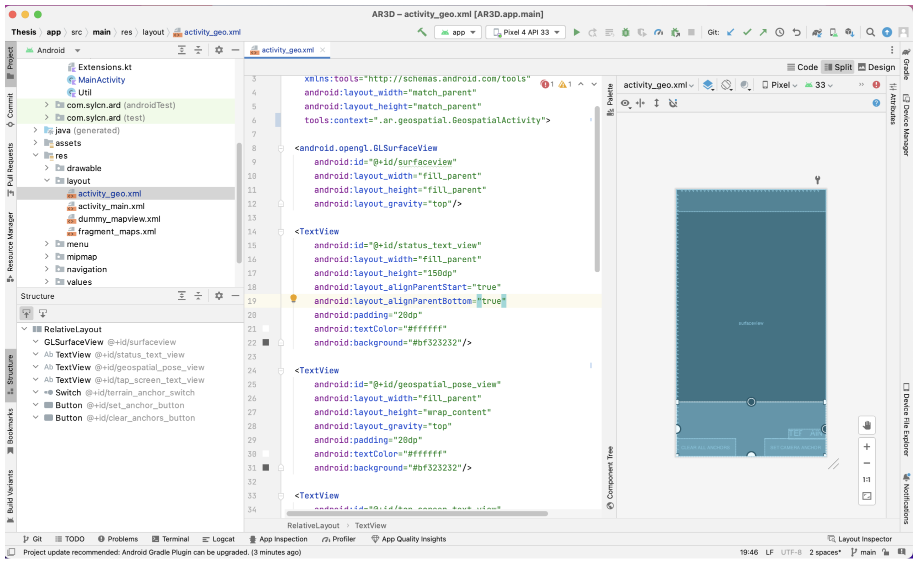
There are three different languages that support the development of Android applications. Kotlin, Java, and C++ languages. Android SDK compiles code along with any data into an APK or an Android App Bundle.

An Android package, which is an archive file with an .apk suffix, contains the contents of an Android app that are required at run time and it is the file that Android-powered devices use to install the app.

An Android App Bundle, which is an archive file with an .aab suffix, contains the contents of an Android app project including some additional metadata that is not required at run time. An AAB is a publishing format and is not installable on Android devices, it defers APK generation and signing to a later stage. [51]

3.2.1 Kotlin

Kotlin is an open-source programming language developed by JetBrains. In late 2017 Google announced that they will conduct Android development with Kotlin officially. In Google I/O 2019, the Google Android development team has also announced that developments will be increasingly Kotlin-first. [53]



```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ar.geospatial.GeospatialActivity">

  <android.opengl.GLSurfaceView
      android:id="@+id/surfaceview"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="top"/>

  <TextView
      android:id="@+id/status_text_view"
      android:layout_width="fill_parent"
      android:layout_height="150dp"
      android:layout_alignParentStart="true"
      android:layout_alignParentBottom="true"
      android:padding="20dp"
      android:textColor="#ffffff"
      android:background="#bf323232"/>

  <TextView
      android:id="@+id/geospatial_pose_view"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_gravity="top"
      android:padding="20dp"
      android:textColor="#ffffff"
      android:background="#bf323232"/>

  <TextView
```

Figure 7: Android Studio layout editor interface.

There are four main reasons to select Kotlin over Java. Kotlin is expressive and concise, it brings safer code which means apps developed with Kotlin are less likely to crash. Interoperability is another rich feature of Kotlin. It is 100% consistent with the Java programming language. It has also structured concurrency which brings background task management easy with the help of coroutines. [53]

3.2.2 Android Studio

Android Studio is the official IDE for Android Development. The main purpose behind Android Studio is to accelerate developer efficiency and help developers build high-quality apps. It's based on IntelliJ IDEA by JetBrains.

Android Studio offers many features popular in other IDEs such as pushing code and resource changes promptly while having a running app, intelligent code editor for syntax highlighting functionality, and code completion. It also has an excellent Emulator which powers when running an app and provides a faster way of deployment instead of real device testing. The simulation could be done not only for smartphones but also for other device types such as smart watches, android autos, and tablets.

As shown in Figure 7, Android Studio has a powerful tool called layout editor. Visual elements of any Android application as well as their relation to other elements could be created by the layout editor. It gives the ability to see developed applications before compiling them. [54]

3.2.3 Android Components

There are four main components in Android Development. Activities, Services, Broadcast Receivers, and Content Providers. **Activities** are the main entry point that gives the user the ability to interact with the app. It represents a single screen with a user interface. **Services**, created for the purpose of background and long-running operations while the app is working. **Broadcast Receivers** aims to deliver events to the app while outside of the activity lifecycle. It could be exampled like getting an SMS for verification, getting notification, etc. Last but not least, **content providers**, as understandable from its name, manage communication between apps. Through the content provider, other apps can query or modify data if the content provider allows it. For instance, the Android system provides a content provider that manages the user's contact information. As such, any app with the proper permissions can query the content provider [51]

3.3 Android App

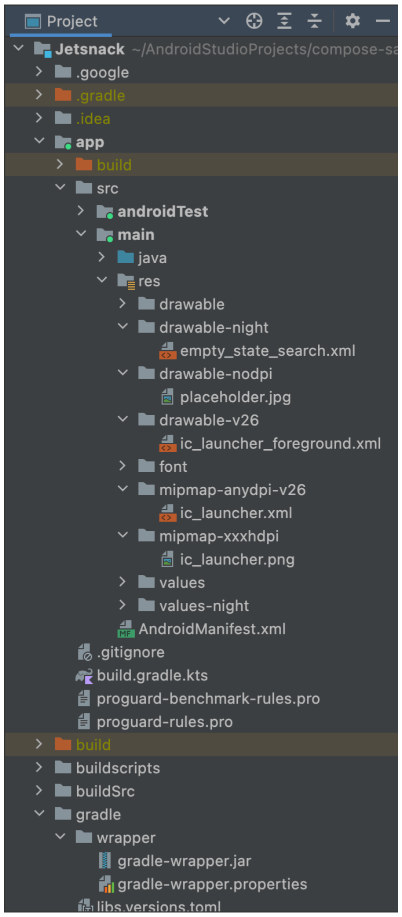
In this section, brief information regarding a simple android app will be explained according to its components. As explained in the section on Android Studio, when developing an Android app, Android Studio should be used as per recommendations by Google. In order to create a new Android app, the following steps are required. A template should be selected as soon as creating a new project. This will bring a screen to give the main properties of the app. They can be explained in the following list;

- **Name:** The app name will be visible on the phones.
- **Package name:** The unique identifier for an Android application. It is used to identify the app's code files, resources, and assets.
- **Minimum SDK:** The minimum level of Android API (Application Programming Interface) that the app requires to run. It specifies the lowest version of Android on which the app can be installed and run without any issues. When selecting the minimum SDK, Android Studio informs its users to select the best version by providing up-to-date ranges.

After that information has been entered, the empty app is opening with the desired template. The minimal app contains two different components called, app and gradle. In order to view those files in advance, project view should be used as shown in 8.

when opening the project view, a lot more files and directories will be visible to the developer, those files and directories include the following files;

- **build:** Contains build output for the app.
- **libs:** Holds private libraries that are added to extend the functionality of the app.



```
Project
v Jetsnack ~/AndroidStudioProjects/compose-sa
  > .google
  > .gradle
  > .idea
  v app
    > build
    v src
      > androidTest
      v main
        > java
        v res
          > drawable
          v drawable-night
              empty_state_search.xml
          v drawable-nodpi
              placeholder.jpg
          v drawable-v26
              ic_launcher_foreground.xml
          > font
          v mipmap-anydpi-v26
              ic_launcher.xml
          v mipmap-xxxhdpi
              ic_launcher.png
          > values
          > values-night
          AndroidManifest.xml
    .gitignore
    build.gradle.kts
    proguard-benchmark-rules.pro
    proguard-rules.pro
  > build
  > buildscripts
  > buildSrc
  v gradle
    v wrapper
        gradle-wrapper.jar
        gradle-wrapper.properties
    libs.versions.toml
```

Figure 8: Android app project files [55]

- **src:** Contains all code and resource files in the project.
- **androidTest:** Contains instrument test codes on that run on Android devices or emulators.
- **cpp:** Native C and C++ code place that is used to develop native Android applications which are low-level android apps that target physical device components, such as sensors and touch inputs.
- **main:** A place for the sources of Android apps that are shared by all build variants.
- **AndroidManifest.xml:** Every app must have that file as the root of the project structure. This file describes essential information about an Android app's build tools, the operating system, and Google Play. The manifest file is handling the following points: The components of the app, which include all activities, services, broadcast receivers, and content providers. The permissions that the app needs in order to access protected parts of the systems. The hardware and software features that the app requires, such as a camera, and open gl. etc.
- **java:** All java and kotlin classes are staying in this direction.
- **res:** Application resources, such as drawable files, layout files, and UI string files are the content of this inventory.
- **assets:** Contains files to be compiled into an APK file as-is. The best location for textures.
- **test:** Contains local test codes.
- **build.gradle (app):** This file specifies build configurations for app-level purposes.
- **build.gradle (project):** A project-level build configuration when the app is containing more than one module. The configuration here applies to all modules

3.4 ARCore

ARCore is a powerful tool for building augmented reality (AR) apps. It has three key capabilities to interactive virtual content with the real world:

- **Motion tracking:** The phone understands and tracks the user's position. To understand how the motion tracking world, ARCore uses a process called simultaneous localization and mapping (SLAM) which is ARCore uses the captured photos as feature points and uses these points to compute changes in location.

- **Environmental understanding:** Allows phone to detect location; horizontal, vertical, and angled surfaces. This feature is achieved by feature points that appear to lie on surfaces. For example, tables, and walls.
- **Light estimation:** As understandable from its name, allows apps to estimate the environment's light conditions. [56]

Apart from these three main components, ARCore is providing other features. Firstly, in hit testing, ARCore can take an (x,y) coordinate from the phone's screen and project a ray into the camera's view of the world. This returns any geometric planes or feature points that the ray intersects, along with the pose of that intersection in world space. This allows users to select or interact with objects in the environment. Secondly, oriented points let to place virtual objects on angled surfaces. When the user performs a hit test that returns a feature point, ARCore looks at nearby feature points and uses those to estimate the angle of the surface at the given feature point. ARCore then returns a pose that takes that angle into account. It's important to note that surfaces without texture, such as a white wall, may not be detected properly, as ARCore uses clusters of feature points to detect the surface's angle.

Moreover, poses can change as ARCore improves its understanding of its own position and its environment. To place a virtual object, the user needs to define an anchor to ensure that ARCore tracks the object's position over time. Often, the user creates an anchor based on the pose returned by a hit test.

Because poses can change, ARCore may update the position of environmental objects like geometric planes and feature points over time. Planes and points are a special type of object called trackables. These are objects that ARCore tracks over time. Anchoring has given the ability to place virtual objects to specific trackables to ensure that the relationship between the virtual object and the trackable remains stable even as the device moves around. This means that if placing a virtual object in an Android figurine on the desk if ARCore later adjusts the pose of the geometric plane associated with the desk, the Android figurine will still appear to stay on top of the table.

Augmented Images is a feature that allows building AR apps that can respond to specific 2D images, such as product packaging or movie posters. Users can trigger AR experiences when they point their phone's camera at specific images. For example, they could point their phone's camera at a movie poster and have a character pop out and enact a scene.

ARCore also tracks moving images, such as a billboard on the side of a moving bus. Images can be compiled offline to create an image database, or individual images can be added in real-time from the device. Once registered, ARCore will detect these images, and the images' boundaries, and return a corresponding pose.

Overall, ARCore is an exciting technology that can help bring AR app ideas to life. With hit testing, oriented points, and trackables, it's promising to create immersive AR experiences that respond to user's interactions with the real world. [56]

3.4.1 Coordinate Systems in AR

Coordinate systems in AR is also crucial subject as combining real world object with virtual object is more challenging question when considering AR applications. AR apps typically use three different coordinate systems: the World Coordinate System, the Camera Coordinate System, and the Screen Coordinate System. The World Coordinate System is a fixed coordinate system that defines the real-world location and orientation of the device. The Camera Coordinate System is a coordinate system that is relative to the device's camera. It moves and rotates with the camera, and its origin is usually at the camera's optical center. The Screen Coordinate System is a 2D coordinate system that corresponds to the device's screen. It is used for user interface elements, such as buttons and menus.

In order to properly align virtual objects with the real world, AR apps need to know the transformation matrix that maps points from the world coordinate system to the camera coordinate. This transformation can be obtained using various techniques, such as GPS, inertial sensors, and computer vision. Once this transformation is known, virtual objects can be positioned and oriented correctly in the camera's view, using the camera coordinate system as shown in Figure 9

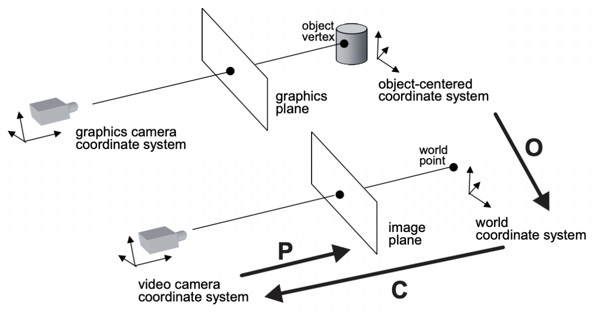

object vertex
graphics plane
object-centered coordinate system
graphics camera coordinate system
O
world point
image plane
world coordinate system
P
video camera coordinate system
C

Figure 9: AR Coordinate Systems.
[57]

3.4.2 Geospatial API

The ARCore Geospatial API gives the ability to the user for attaching any content remotely to any area covered by Google Street View. It brings to provide AR experience on a global scale. Geospatial API uses device sensors and GPS data to detect the environment, then it's matching detected part of the real world. The outcome of this usage is to create a virtual positioning system to place any object remotely within the AR environment as shown in Figure 10.

Global localization with VPS: Street View images from Google Maps were used for global localization to provide geospatial poses which have been captured for more than 15 years. Those images are the fundamental points of the virtual positioning system. The parts of images are combined across tens of billions of them where they have been identified by deep neural networks. After identification, they are combined to compute a 3D point cloud of the global environment. The environment consists of trillions of points and covers nearly all countries. [58]

In the Geospatial API, a neural network is utilized to process pixels received from the user's device, in order to identify recognizable features in the environment. These features are then matched to the VPS localization model, which is used by computer vision algorithms to determine the precise position and orientation of the device. This approach enables the system to provide location data that is significantly more accurate than traditional GPS-based methods. [58]

Placing anchors: In contrast to other ARCore APIs like the ARCore Cloud Anchor API, which require explicit mapping of a space to determine a device's pose, the Geospatial API uses image maps to accurately position anchors without the need for manual mapping. These image maps are global and adhere to the WGS84 specification, providing horizontal (latitude and longitude) and vertical (altitude) position information for anchors. With the Geospatial API, anchors can be easily placed at a specific location on the globe, without requiring manual mapping of the environment. [59]

Terrain anchors: The Terrain anchor, which is a variant of the Geospatial anchor, enables the placement of AR objects solely based on latitude and longitude coordinates while leveraging data from Google Maps to accurately determine the altitude above ground level. [58]

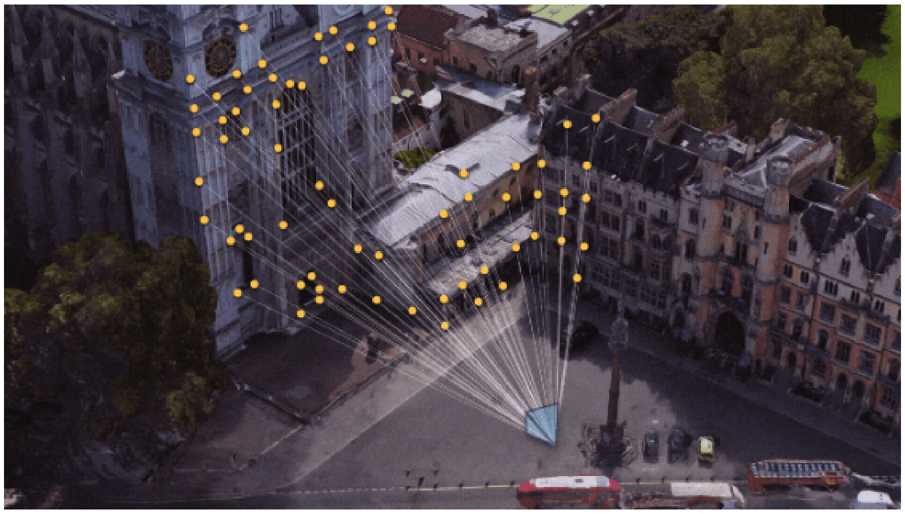


Figure 10: ARCore Geospatial API principles.
[44]

In the context of developing a mobile AR app by considering geospatial capabilities, the user's location is also related to an absolute global frame of reference, typically through the location information provided by GPS coordinates. There are 4 pillars that need to be considered when developing geospatial-oriented AR applications.

- **Camera**: In the field of Augmented Reality, the world, and its surroundings can be understood by users through the use of a camera feed. The camera feed is utilized as a backdrop to integrate augmentations into the user's surroundings. The analysis of the camera feed in real-time is done to identify anchoring patterns or precisely locate features. This enables the user to feel as though the augmentations are a part of their actual surroundings.

- **GPS**: Coordinates are the most accurate information when retrieving the device's location. It will work more efficiently outdoors. Using GPS requires significant power consumption. The ideal usage for geospatial AR applications should have to provide coordinates. In addition to latitude and longitude, the GPS can also provide altitude information. The location provided to AR is the WGS84 coordinate system. The provided altitude is above the reference ellipsoid, not mean sea level.

- **Motion Sensors**: The motion sensors are useful for monitoring device movement, such as tilt, shake, rotation, or swing. The magnetometer, gyroscope, accelerometer, barometer, 3D orientation, and 3D position are components of motion sensors. To explain those sensors, firstly; **Magnetometer** is sensing the Earth's magnetic field to establish the direction of the magnetic north, like a classic compass. The magnetometer, together with the gyroscope and accelerometer, allows the establishment of the 3D orientation of the device. **The gyroscope** sensor reports angular velocity. **The accelerometer** is sensing linear acceleration as the device is moved. **The barometer** is observing atmospheric pressure can be useful to establish much more altitude, based on a reference sea level pressure.

- **Size, weight, and power usage**: Using location updates, sensors, and camera capabilities is power-consuming. Turning off those features when not using them is crucial to improve working duration. Apart from that, rendering images will require Graphical Processing Unit to show a large number of items and/or 3D geometry. All considerations make augmented reality applications rather power consuming, as well as heat and the size of a device can be related to both its ability to dissipate this heat and the capacity of its battery.

4 Methodology

The chapter will be elaborate on components of methodology to explain the requirements when developing Android apps and when developing those kinds of applications which data formats are required, and how the development environment selected.

4.1 Requirements

The goal of this thesis is to integrate CityGML-based building data into an AR environment on a street view level. To achieve this, an Android app will be developed that utilizes OBJ data that has been converted from CityGML or CityJSON using Google's ARCore SDK. The app will leverage the Geospatial API to accurately place the OBJ files in their actual location using latitude, longitude, and altitude information.

The methodology involves several steps. Firstly, the CityGML or CityJSON data will be processed and converted into OBJ files. Next, the ARCore SDK will be used to develop an Android app with augmented reality capabilities. The app will utilize the Geospatial API to accurately place the OBJ files at their actual location. To evaluate the effectiveness of the solution, user testing will be conducted to assess the app's ability to accurately integrate CityGML-based building data into an AR environment on a street view level.This approach provides a new way to visualize building data on a street view level using AR technology. It has the potential to benefit urban planners, architects, and city officials by providing a more intuitive way to understand building data and its impact on the surrounding environment.

The selection of appropriate hardware is crucial for the successful implementation of an augmented reality (AR) system. With the rise of affordable yet powerful smartphones, many AR applications have been designed around them. Modern smartphones come equipped with internal sensors, high-resolution cameras, and displays, making them ideal for the development of a complete mobile AR system. Among the most popular mobile operating systems, Apple's iOS and Google's Android are currently the most widely used. Choosing either of these OS for an AR system offers the advantage of a large user base. With approximately 70% of all mobile devices running on Android, it is particularly attractive for AR development, since it is based on the platform-independent programming language Java and Kotlin, allowing for easy source code reuse. However, the vast number of different Android smartphones on the market can make hardware selection challenging. [60] When selecting a smartphone for an AR system, hardware specifications play a critical role. Modern smartphones come equipped with displays that have at least Full-HD resolutions, which are sufficient for AR systems, taking into account the screen size and viewing distance. In addition, the sensors within the device, such as accelerometers, magnetometers, and gyroscopes, are essential for determining the orientation of the physical device. [47]

The camera is another critical component used for capturing images of the physical world for display on the screen and for internal use in pose estimation. Therefore, high-quality images are necessary, and multiple factors, such as the amount of megapixels, sensor size, and pixel size, are important for image quality. In smartphones, sensor sizes are described as fractional numbers in inches, indicating the type of sensor, and their size is measured along their diagonal. Pixel sizes are measured in micrometers and typically range from 1 to 2 micrometers in modern smartphones. However, increasing the amount of megapixels usually requires a larger sensor, which can limit the size of the device. Furthermore, smaller pixels capture less light, which can affect camera performance in darker conditions. A camera with larger pixels performs better in darker conditions than one with smaller ones. [61]

For this thesis, Xiaomi Redmi Note 8 Pro chosen as smartphones for the development of the AR app, listed in Table 2

Xiaomi Redmi Note 8 Pro	
Category	**Value**
Display Resolution	1080 x 2340 pixels
CPU / GPU	Octa-core 2x2.05 GHz Cortex-A76
Storage / RAM	128GB / 6GB
Camera	64 MP, f/1.9, 26mm (wide), 1/1.72", 0.8µm, PDAF
OS	Android 11

Table 2: Xiaomi Redmi Note 8 Pro Specifications

When developing an Android application, the choice of the development environment is an important factor in determining the overall success of the project. While there are several options available in the market like Visual Studio Code, Eclipse, and even IntelliJ Idea, Android Studio has emerged as a popular choice for Android developers due to its versatility, ease of use, and rich set of features. One of the main advantages of Android Studio is its integration with Google's official Android SDK, which provides a comprehensive set of tools and libraries for building high-quality applications. This integration makes it easier for developers to create applications that adhere to best practices and follow the latest Android design guidelines.

Android Studio also offers a powerful code editor with advanced code completion, highlighting, and analysis features, making it easier for android developers to write efficient and maintainable code. The built-in layout editor provides a satisfying interface for creating layouts, which can then be customized using code if necessary. Another key advantage of Android Studio is its support for version control systems like Git, which allows developers to manage their codebase and collaborate with team members more effectively. The integrated debugging tools also make it easier to track down and fix errors during development, reducing the time required for testing and bug fixing.

When compared to other development environments like Eclipse, Android Studio is often seen as a more modern and compelling option, with better performance and a more intuitive user interface. This is reflected in its popularity among developers, with many choosing to use Android Studio for their Android development needs. Due to the listed requirements and powerfulness, Android Studio was selected as the main development environment. [62]

In order to show 3D building data in a street-level AR environment. There must be an AR library to catch this functionality. Since the Android platform was selected as a development platform, the best AR library would be ARCore from Google. [63]

In order to use ARCore in Android applications, the app should target Android phones running Android 7.0 (Nougat) and later. [64]. Although ARCore is supporting a broad range of AR applications, the integration of CityGML-based building data into an AR environment requires a high level of localization that ARCore primarily not supports. Following the explained reasons in the background section, the ARCore Geospatial API was selected.

4.2 App Structure

The purpose of this section is to provide an overview of the architecture and technologies used in the development of the Android app, which aims to integrate CityGML-based building data into an AR environment at the street view level. The app was developed using the Model-View-ViewModel (MVVM) architecture, which separates the app's data, user interface, and logic. This design pattern is helpful for building scalable and maintainable apps that are easy to test and update. The app's data is stored in an OBJ file format, which is converted from CityGML and CityJSON using the FME Quick Translation program. This format is ideal for the app's use case, as it is lightweight and easy to work with in an AR environment. The app also uses Google Maps for mapping purposes, which provides users with an accurate representation of their surroundings. The app's user interface is built using Activities and Fragments. Activities are the app's main entry points, representing the various screens that users interact with. Fragments are reusable components that can be combined with other Fragments to create complex user interfaces. The app's interface is designed to be simple and intuitive, allowing users to easily navigate and interact with the app's features. The ViewModel is responsible for managing the app's data and logic. It provides a clean interface between the Model and View and ensures that the app's data is presented in a consistent and easy-to-understand way. The ViewModel also uses the ARCore SDK for the app's AR capabilities, which allows users to see 3D building data in a real-world environment. The ARCore Geospatial API is used for geospatial processes, such as posing the 3D building data in the AR environment. The app was developed with a minimum SDK version of 24, which is required for the use of the ARCore SDK. This ensures that the app is compatible with a wide range of devices and can provide users with an immersive AR experience as shown in the figure

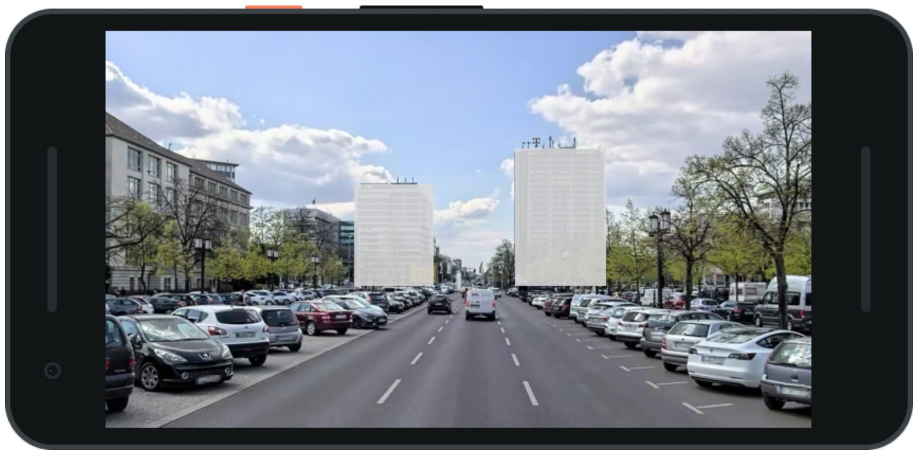


Figure 11: Concept image of desired application
[44]

The implemented AR system can be split into three core components, the augmented world, map view, and physical world, which also define the three pillars of AR in general. To realize the building data in general, there is a need for AR View. This view will act as the main point of interaction in the app as shown in Figure 12 It will basically rely on the permission of users because of the requirements that AR systems require that are described in the Background section.

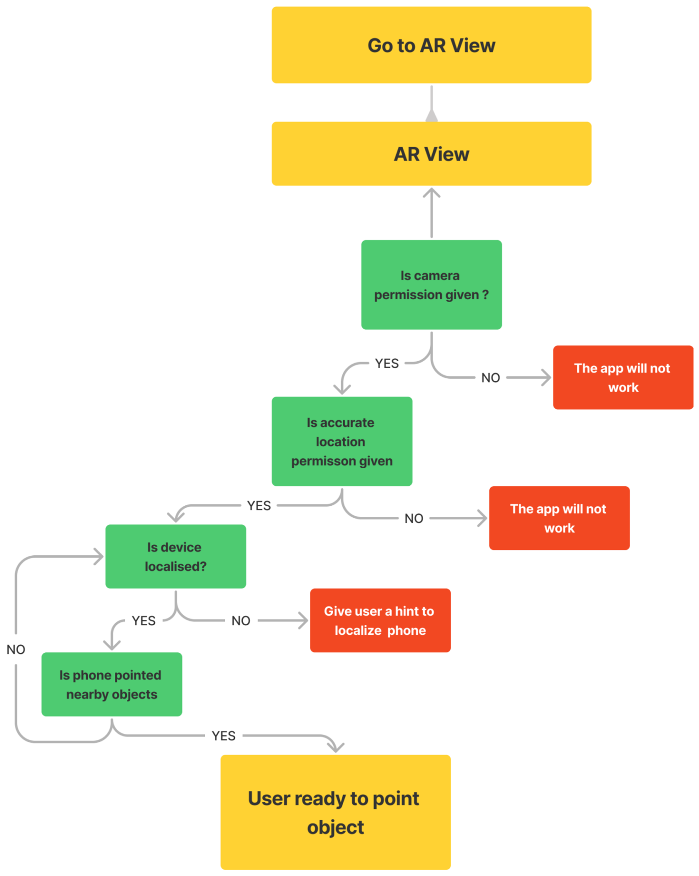

Go to AR View
AR View
Is camera permission given ?
YES
NO
The app will not work
Is accurate location permisson given
YES
NO
The app will not work
Is device localised?
YES
NO
Give user a hint to localize phone
NO
Is phone pointed nearby objects
YES
User ready to point object

Figure 12: AR app workflow.

4.3 Study Area & Data

The testing with ARCore Geospatial API requires testing outside since it's using global localization with VPS. This term could be understood that Street View images from Google maps are being captured for more than 15 years. That's why those images foundation of VPS. Deep neural networks interpret those images and compute the 3D point cloud of the global environment. Since testing outside gives precise results. The study area was selected as the main building of Technische Universitat Berlin which is located at Straße des 17. Juni 135, 10623 Berlin as shown in Figure 13.

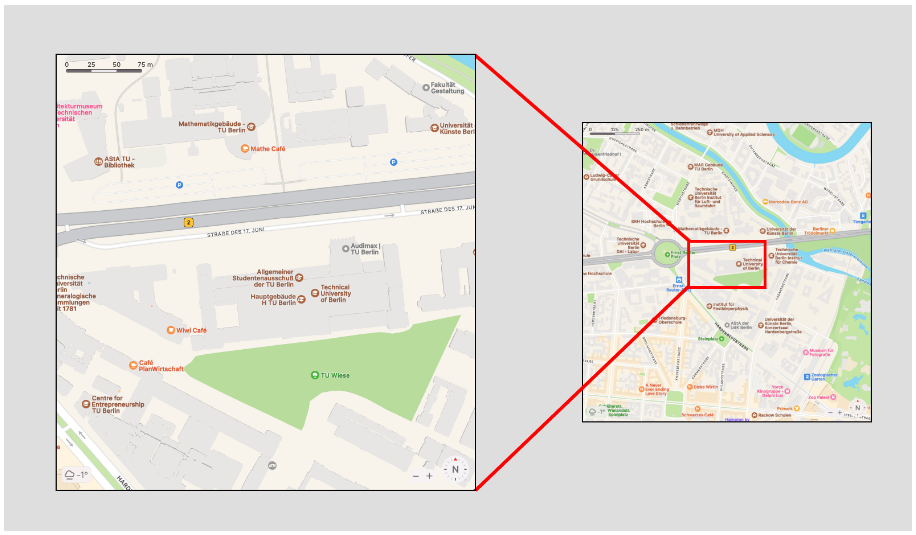


Figure 13: TU Berlin main building map
[44]

The data has been taken from the Berlin 3D - Download Portal which provides open data since 2009. The platform gives the ability to download both the mesh model and the building-oriented LoD2 model. The differences between the mesh model and the building-oriented model could be found in the table. [65] For mobile application development processes, the CityGML LoD2 model has been downloaded and processed.

4.3.1 Data Conversion

OBJ is a common 3D format that has wide software support in 3D modelling and 3D visualisation software. However, OBJ lacks the semantics that is provided in CityGML and CityJSON. In that sense, conversion to an OBJ file is beneficial to take advantage of the wide range of 3D software. [46]

CityGML, CityJSON, and OBJ formats are the main components of the developed app which will be data for visualization of 3D buildings integrated

	3D mesh model	Building-oriented LoD2 model
Validity	Based on a photo flight from August 2020	Initial version 2007-2009, last updated in 2014 based on a photo flight from summer 2013
Characteristics	Textured, comprehensive 3D mesh model without building information	More than 560,000 textured individual buildings based on ALKIS floor plans with roof structures (LoD2 model)
Original file format	OBJ and skins (tiled)	CityGML and skins
Download	Tile by tile as OBJ per interactive selection in the download portal	By district as CityGML (as a ZIP archive) or per interactive selection in the download portal in various formats

Table 3: Differences between 3D mesh model and LoD2 Model

into an AR environment on a street view level. In order to use those data, CityGML data has been converted to CityJSON and OBJ format. Both formats will be observed according to their reliability and performance.

In order to convert those data to a more AR-understandable format a conversion tool called FME | Data Integration Platform will be used. This platform enables conversion from CityGML and CityJSON to OBJ with the FME Quick Translator tool. [66]

When working with the conversion of data between the OBJ and CityGML formats, there are several challenges that need to be considered.

1. OBJ datasets typically consist of sets of polygons or triangles, while CityGML geometry is based on the types defined in the Geometry Markup Language (GML). This means that CityGML supports more advanced concepts, such as polygons with holes and solids, which are not present in OBJ. As a result, a single surface in CityGML may be represented by multiple faces (triangles) in OBJ. [46]

2. CityGML supports thematically differentiated objects and surfaces, which is not the case with OBJ. In OBJ, there is no standard way to differentiate objects or surfaces based on attributes. In contrast, CityGML provides a way to differentiate objects and surfaces based on their thematic attributes, making it more suitable for complex urban models. [46]

3. OBJ has limited support for attributes, whereas CityGML allows several attributes to be assigned to objects or their parts. This means that CityGML can provide more detailed information about objects and their attributes, such as material properties, thermal behavior, and energy performance, which are critical for simulating urban environments accurately. [46]

5 Implementation

The realized mobile app development implementation derived from the requirements as the thesis outcome is described in detail in the following. First, the mobile app architecture will be shown and explained shortly. Other parts of the solution are then described in more detail.

5.1 Design

The first phase of mobile application development is design as its first place the user gets an impression. While designing the mobile application, user experience was taken into account and the design was made by the material design principles determined by Google. [67] According to these principles, the areas accessible to the user in mobile application usage decrease as you go upwards. For this reason, the main functions of the application will be summarized below a design has been made in such a way as shown in Figure 14 [68]

As requirements summarize the needs for the intended app, the app has 4 screens in advance; Splash screen, map screen, add file screen, and AR screen. For those ideas in mind, a proof of concept design as shown in Figure 15 made by Figma which is a tool for designers and developers to visualize their ideas. [69]

5.2 General App Architecture

General recommended app architecture can be explained using Model, View, and ViewModel. [5] To explain those components deeper model holds data, and ViewModel observes those data to expose to the View. As explained in the methodology section, the general architecture will be as follows. Android Jetpack libraries will be used for handling interaction processes. Google Maps SDK will also be used for mapping purposes and ARCore Geospatial API will be used for AR interactions.

To explain project architecture deeply, the whole app is divided into four different packages which mean the app is maintained as a package by features as shown in Figure 16. To explain those app structure elements independently; **ar** section is a package for whole classes in order to catch AR functionality.

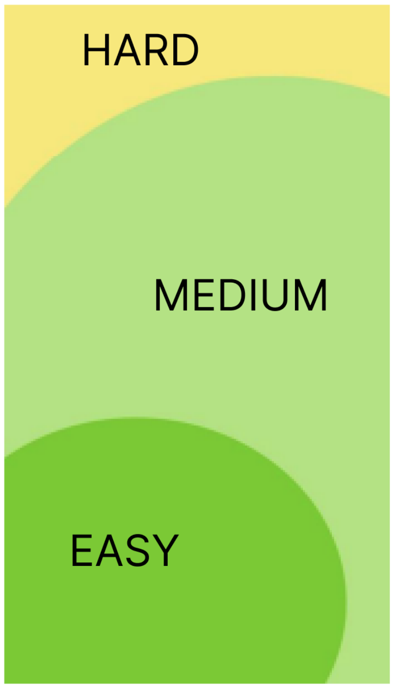


Figure 14: App UX design approach.

helpers package is for the ARCore library which is including core Java classes that are maintained by Google. **map** package is for the map screen as the main screen of the app. **renderer** package is also for ARCore's object renderer coming from the library. Apart from those points, the app has **a manifest** section about maintaining activity properties such as theming, permission, etc. **assets** for storing 3D objects. **res** folder for the resource purposes such as storing dimensions, layouts, strings, etc. The Gradle scripts are also for the configuration of the project.

5.3 Development of the app

The general development structure is shown in Figure 17. As explained in the requirements section the development structures have 3 different phases, selection of libraries, preparing views, and adding business logic to the app. The planned approach for the development of the app will be converting data from CityGML to OBJ format and showing them using the loader class.

5.3.1 Dependencies & Libraries

The dependencies and libraries are the main components of the app when considering development. The following dependencies have been used to develop

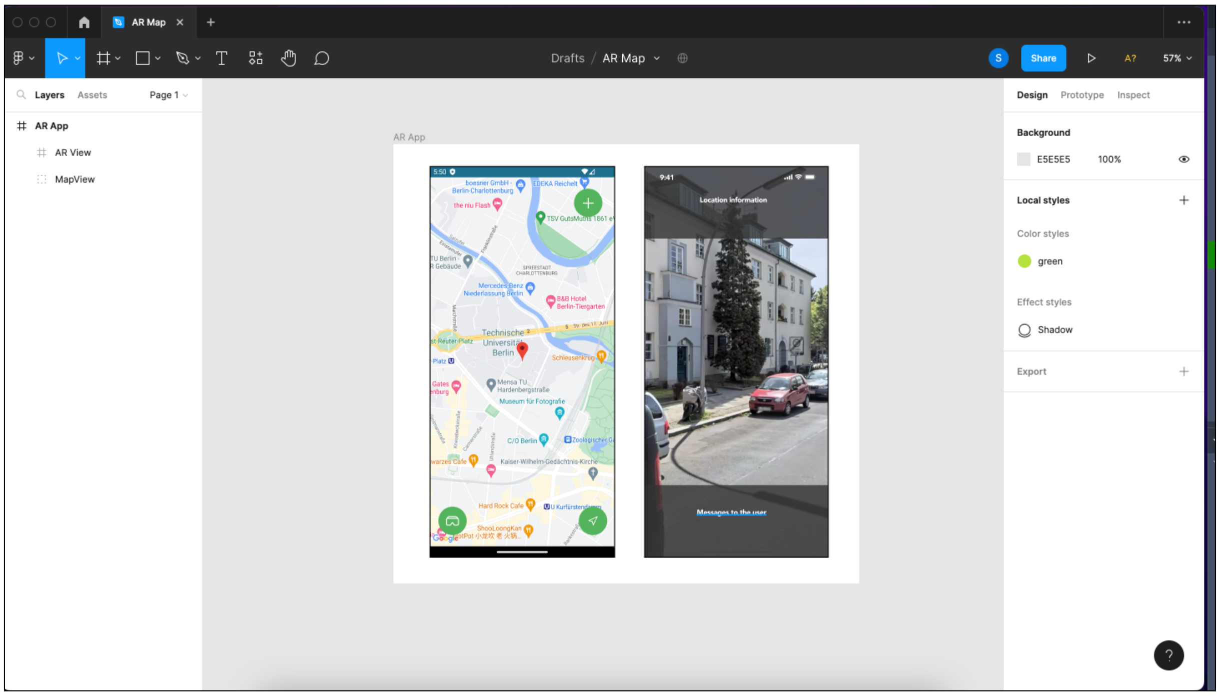
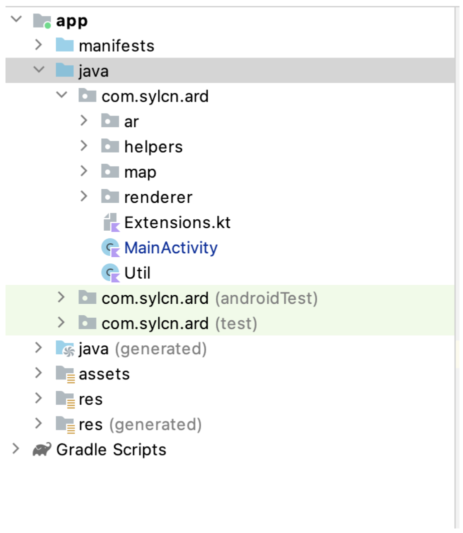


Figure 15: Proof of concept design with Figma

desired Android application. Some of those libraries are coming in a default way which are Android core, AppComppat, Material, ConstraintLayout, Junit, and Espresso. All libraries used in the app are explained below;

- **Android Core:** Provides a set of Kotlin extensions for core Android framework APIs.
- **Android AppCompat:** Ads support for the Action Bar user interface design pattern.
- **ConstraintLayout:** Allows to create large and complex layouts with a flat view hierarchy.
- **Android Core Splash Screen:** Provides a simple way to show a splash screen on app startup.
- **Android Activity KTX:** Set of Kotlin extensions for the Android Activity class.
- **Android Fragment KTX:** Set of Kotlin extensions for the Android Fragment class.
- **Android Navigation Components:** Support for fragment navigation.



```
app
  manifests
  java
    com.sylcn.ard
      ar
      helpers
      map
      renderer
      Extensions.kt
      MainActivity
      Util
    com.sylcn.ard (androidTest)
    com.sylcn.ard (test)
  java (generated)
  assets
  res
  res (generated)
Gradle Scripts
```

Figure 16: General app structure

- **Android Navigation UI:** A library supports for the Navigation component, including support for the app bar, navigation drawer, and bottom navigation.
- **Google Play Services Auth:** Google Sign-In and authorization which used to communicate Google Cloud.
- **Google Play Services Location:** Bring location-based services, including geofencing and activity recognition to the Android Apps.
- **Google Play Services Maps:** In order to use Google Maps API, this library should be added.
- **Lifecycle Common Java 8:** Lifecycle aware components.
- **Concurrent Futures:** Small library to support concurrent programming in Java when rendering objects.
- **Google Filament Android:** A real-time physically-based rendering engine.
- **Google AR Core:** Augmented Reality development for native android apps.
- **Google Material:** Material Design components and guidelines.
- **Google Guava:** Provides a set of core libraries for Java programming.
- **OBJ:** Support for loading OBJ models.

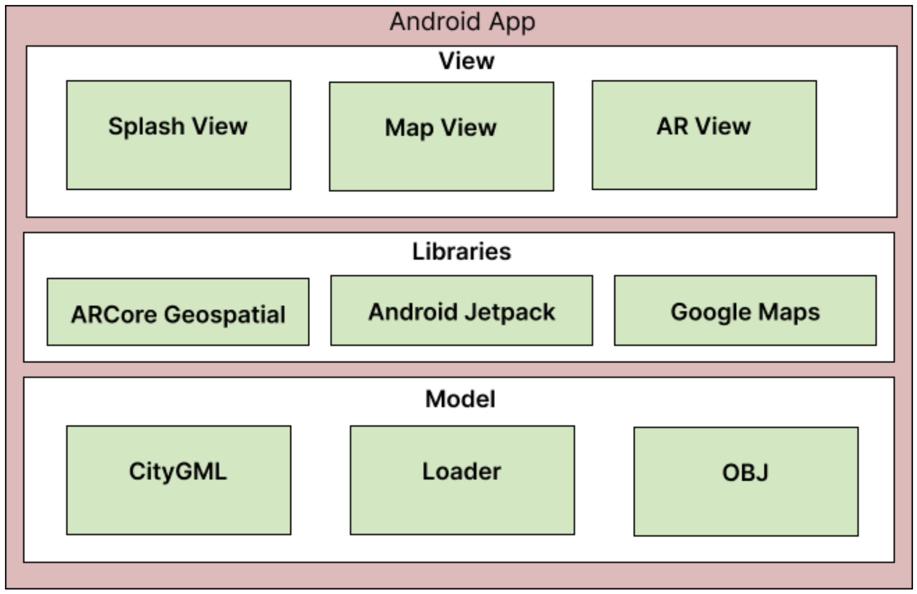


Figure 17: General development structure components

- **JUnit:** Testing framework for Java programming.
- **Android Test Ext JUnit:** Support for JUnit testing on Android devices.
- **Espresso Core:** Provides a set of testing APIs for UI testing on Android.

5.3.2 Splash Screen

The first implementation screen is to create a splash screen. The main purpose of the splash screen is to show the user a screen while the app is being prepared. For the intended application, a splash screen API has been used that fits every version of the Android phones. [70] Due to splash screen created using Google's library provided previous section it's maintained by manifest file. No separate file class is not generated for the splash screen purposes.

5.3.3 Map Screen

In order to develop the map screen feature a `MapFragment` created under `map` package. User needs to be directed to the MapView as soon as app is open. The app's navigation flow handled by `MainActivity` which is the main point for the screen. To direct the user to the `MapView`, a navigation XML created that includes `MapFragment` as a destination. Fragment is the component of Android Development provides flexibility to create responsive screens on top of activites. Their functionality depends on Activity Lifecycle as well as having separate lifecycle. From implementation of AR app, two main method overrided to use Fragments. `OnCreateView` is called to have the fragment instantiate its user interface view and `OnViewCreated` is being called immediately after `onCreateView` has returned, but before any saved state has been restored in to the view. [71]. To provide mapping functionality in the `MapView` Google Maps Android SDK was used. Apart from that the **map screen** has different use cases;

- **Main View:** The main map view is opening with a marker showing TU Berlin as a marker and has a fresh map initiated. The user has a chance to pan, zoom and pinch in the map view. This View has been achieved by listening the `OnMapReadyCallBack` from Google Maps library. If map is ready, then it means we can do processes to make data manipulation Google Maps. In that case, as soon as user opens the app, map loads and then marker is loading in order to be show desired point with marker added.

- **Add file:** This button allows the user to add CityGML and CityJSON to the app whenever selecting it from the file system. In order to work with this functionality, the user must have to give consent to the app for giving permission to read their external storage. Otherwise, working with the files is not possible. The reading files is achieved by parsing XML or JSON parsers.

```
private fun openFile() {
        val intent = Intent(Intent.ACTION_OPEN_DOCUMENT).apply {
            addCategory(Intent.CATEGORY_OPENABLE)
            type = "application/gml"
        }

        startActivityForResult(intent, pickPdfFile)
    }
```

- **Go to the location:** This allows the phone to show the user's location on the map by clicking the button. The location-showing process also requires permission from the user. Starting from Android version 6.0, runtime permission should have to be requested. [72] In the developed app, runtime permissions are also requested as soon as the user clicks the app. User should give the permission for location purposes. If user grants desired location permision, a marker could be added by following code pieces. The code piece is working by tapping `My Location` button in the app.

```
showMyLocationButton.setOnClickListener {
            if (!LocationPermissionHelper.hasFineLocationPermission(requireA
                LocationPermissionHelper.requestFineLocationPermission(requi
            } else {
                //go to user location
                mainMap?.isMyLocationEnabled = true
                mainMap?.uiSettings?.isMyLocationButtonEnabled = false
                getLastLocation()
            }
        }
```

- **Go to AR View:** This feature allows the user to open AR View by controlling whether the user has AR supported-device or not. If the user doesn't have an AR-supported phone the AR View functionality is not directing the user to that view. Since the AR View is maintained by an Activity. Navigation done with Intent class of Android which is being used for launcing Acitivies mainly.

5.3.4 Add File Screen

The main idea to create this screen is to add CityGML/CityJSON data from the phone's file system, then convert this data to an OBJ file to show in the AR environment. The selected CityGML/CityJSON file also be shown on the map as a point marker to show users which location they are dealing with. Due to lack of conversion from CityGML and CityJSON to OBJ file, more software oriented way used to achieve this functionality. On the other hand, parsers created for parsing CityGML and CityJSON. When creating parsers, following considerations have been taken into account.

The first step in the parsing process is to read in the CityGML or CityJSON data using a parser library. There are several parser libraries available for both CityGML and CityJSON formats, such as CityGML4j, GeoTools, and Jackson. The parser libraries will generate a data structure that represents the parsed CityGML or CityJSON data. Once the CityGML or CityJSON data is parsed, the next step is to convert it to 3D geometry data. This involves extracting the relevant information from the parsed data, such as building footprints, heights, and textures. The geometry data can then be stored in an appropriate format, such as OBJ file.

5.3.5 AR View Screen

The most functional screen of the app is called AR View which aims to interact users in real world. To explain this screen in detail; when the user opens the AR view, the user has been warned of privacy concerns and camera permission is requested from the user in order to interact with the real world using the rear camera. If the user grants the permission then the accurate location permission is being requested by the app because ARCore Geospatial API works just with accurate location information [59]. As soon as all required permissions are given by the user, then localization phase begins which is using the location of the phone to understand at which point the user stand.

Additionally, if all localization processes went well, virtual positioning takes part in the application posing. In order phone to create VPS, the phone must be located in nearby objects, such as buildings, stores, etc. ARCore Geospatial API is using Google StreetView images and estimates the user's location with neural network algorithms. [58]

Last but not least, after determining the user's location and device geospatial pose, objects could be placed by touching the screen. The app also has two different functionalities one setting geospatial anchors and the other one setting terrain anchors. After setting those anchors in the AR world using VPS, the object is anchored in its position spatially. [73]

In order to achieve previously explained steps, in the implementation side `ar`, `helpers` and `renderer` packages used.

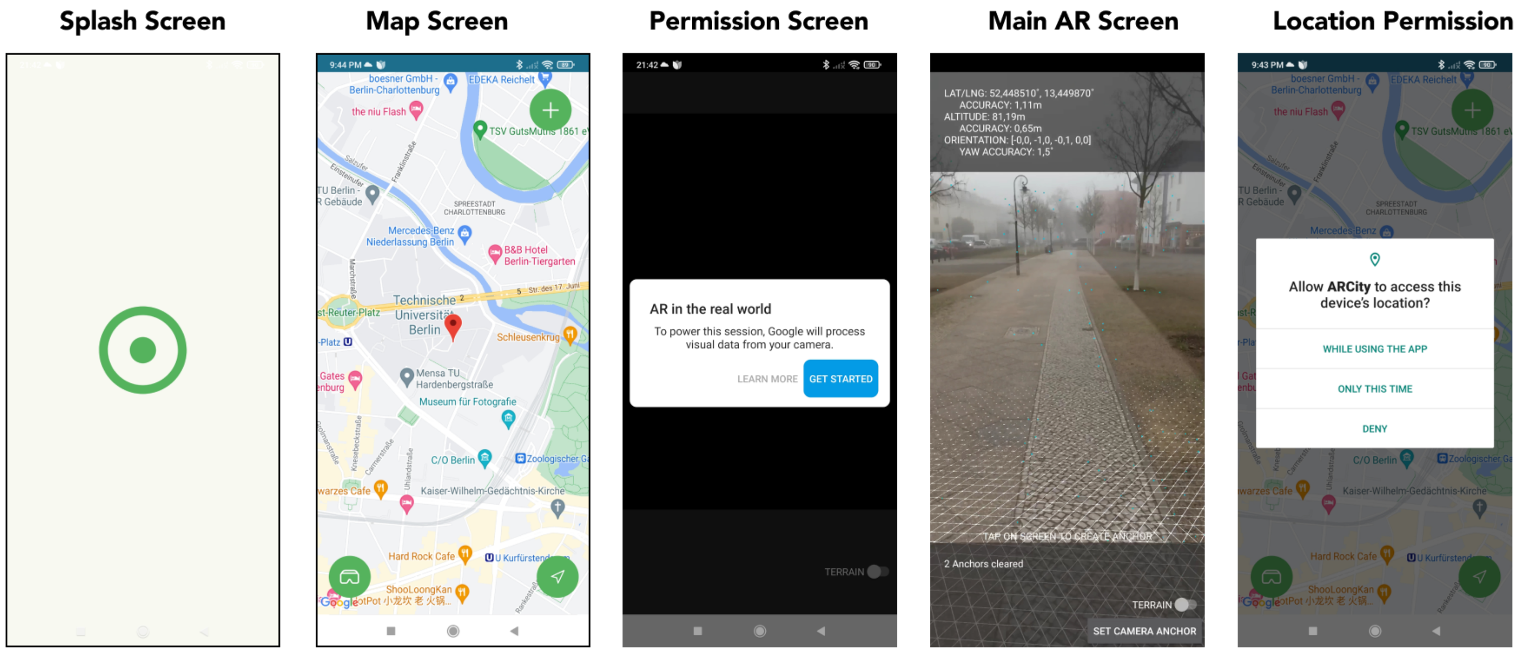
`ar` package has GeospatialAcitity to responsible for main AR View. PrivacyNoticeDialog fragment to give user information about privacy concerns. VpsAvailabilityNoticeDialogFragment is to give user availability of VPS. Those privacy and VPS dialogs are provided from Google. [74]

`helpers` package includes classes coming from the ARCore SDK to use functionality of Augmented Reality such as a session lifecycle helper which manages an ARCore Session using the Android Lifecycle API. Before starting a Session, the class requests installation of Google Play Services for AR if it's not installed or not up to date and asks the user for required permissions if necessary. Camera permission helper to ask camera permissions. Depth settings to manage the occlusion option setting and shared preferences. Display router helper to track the display rotations. Full screen helper to set up the Android full screen mode. Instant placement settings to get functionality of instant placement. Location permission helper to ask for user's location. Map Touch Wrapper to get finer granularity when tapping on GoogleMap views by interpreting touch events. Snackbar helper to give user information as soon as their processes finished or dismissied. State helper to give user meaningful information about outside regarding the VPS and GPS state.

`renderer` packages includes classes for rendering purposes. Background renderer for AR Camera background, plane renderer, specular cubemap filter which filters a provided cubemap. Framebuffer associated class with texture. GLError for handling OpenGL errors. Mesh which is a class for collection of vertices, faces, and other attributes that define how to render a 3D object. Shader and Texture classes for showing 3D object in an AR world.

6 Results

As depicted in Figure 12, an Android app, having CityGML-based building data integrated into an AR environment on a street view level has been developed. The tests have been made with Xiaomi Redmi Note 8 Pro device. However, ARCore supports only 3D object formats such as.OBJ, .fbx,.psd. The development showed that direct usage of CityGML or CityJSON is not possible due to the capabilities of the Google ARCore library. [75]



AR in the real world

To power this session, Google will process visual data from your camera.

LAT/LNG: 52,448510°, 13,449870°
ACCURACY: 1,11m
ALTITUDE: 81,19m
ACCURACY: 0,65m
ORIENTATION: [-0,0, -1,0, -0,1, 0,0]
YAW ACCURACY: 1,5°

2 Anchors cleared

Allow **ARCity** to access this device's location?

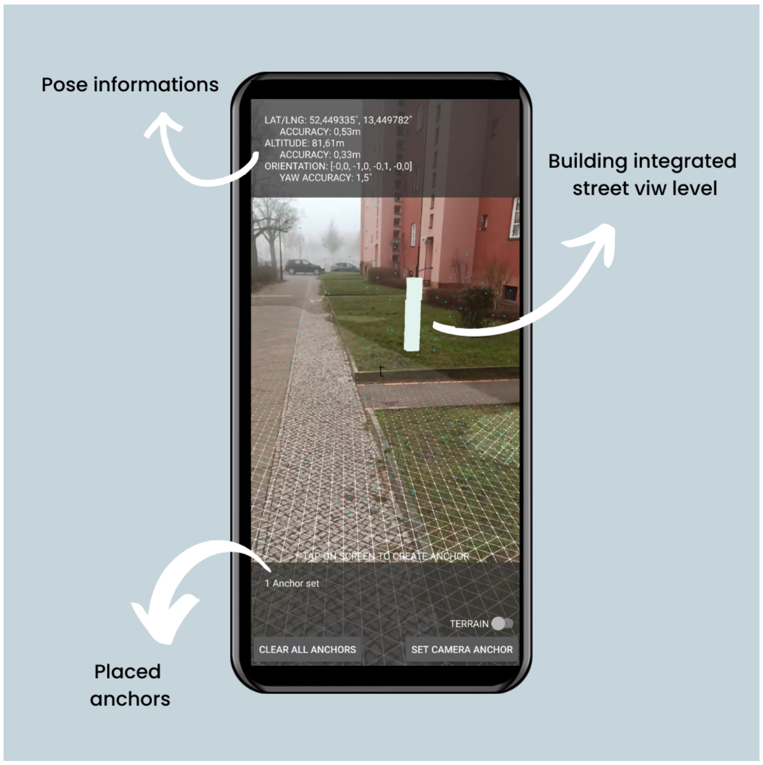
Figure 18: 3D AR app using ARCore Geospatial API.

Additionally, the OBJ file converted from the CityGML file has also been shown at the street view level but the challenge of showing this 3D object at a street-view level requires advanced techniques of posing which is related to visual programming such as shader language.

The intended use cases for runtime permissions have already been achieved as users are warned about privacy, their file system permission, camera permission as well as their location permission. If the user didn't grant these permissions, the app won't work.

The conversion from CityJSON data experimented with FME desktop application but the process turned out to be useless because converting from CityGML to OBJ seems to give the expected results as shown in Figure 19. To explain this figure, the small building is far away from the user's position that's the reason it's rendered as a small object. When moving towards to this building object in the AR environment, it's getting bigger but as stated before, it lacks its real positioning.

Testing of the app is also challenging, as it was required to test outside because Google ARCore Geospatial library is depending on Google Street View images, that's why a full test could be possible just by going outside. [58]


```
LAT/LNG: 52,449335°, 13,449782°
    ACCURACY: 0,53m
ALTITUDE: 81,61m
    ACCURACY: 0,33m
ORIENTATION: [-0,0, -1,0, -0,1, -0,0]
    YAW ACCURACY: 1,5°
```

TAP ON SCREEN TO CREATE ANCHOR

1 Anchor set

Figure 19: Integrated 3D building data into AR

7 Conclusion and Future Work

The development of the app shows that mobile devices, such as smartphones, are sufficient to realize a performant mobile AR app for showing building information with CityGML data. Also, the development process shows that posing the building to its location accurately requires a high level of algorithms, for instance, a quaternion-based visualization. [18]

For future work, the app could be extended features below;

- Instead of a vector map that the app is supporting to show map information to the user, a satellite imagery map could be added to show users detailed information about their location.
- A more automated way of data conversion could be done on both the client and server sides. On the client side, the conversion from CityGML to OBJ could be achieved.
- A more lightweight data format, such as CityJSON could be used as a source of conversion data which will give developers to implement features easily as CityJSON is more lightweight than CityGML. [76]
- Server driven approach could be added to a recent application which is to hold data in the server and show those data to the user according to the user's position. This method will require to implementation of some pagination options for the client to show buildings in a performant way. For example, the data from the server is requested as soon as the user reaches the area of visualization and is shown to that user.
- Server driven approach could be added to a recent application which is to hold data in the server and show those data to the user according to the user's position. This method will require to implementation of some pagination options for the client to show buildings in a performant way. For example, the data from the server is requested as soon as the user reaches the area of visualization and is shown to that user.

- The resulting app could be achieved not only using native approaches but also using game engines, such as Unity, and Unreal Engine. Those game engines allow users to create realistic 3D environments containing terrains as well as objects. Using those platforms will also be beneficial to create an app from one platform and the outcome will be providing a running app to both mobile platforms, such as iOS and Android. [77] [78]

References

[1] Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova, and Arzu Çöltekin. Applications of 3D City Models: State of the Art Review. *IJGI*, 4(4):2842–2889, December 2015.

[2] Ahmed Kareem Jebur. Application of 3D City Model and Method of Create of 3D Model- A Review Paper. *Saudi J Civ Eng*, 6(4):95–107, April 2022.

[3] Mehmet Buyukdemircioglu and Sultan Kocaman. Reconstruction and Efficient Visualization of Heterogeneous 3D City Models. *Remote Sensing*, 12(13):2128, July 2020.

[4] Ma Leticia Jose C. Basilan, https://orcid.org/0000-0003-3105-2252, Maycee Padilla, and https://orchid.org/0000-0001-5025-12872, maleticia-jose.basilan@deped.gov.ph, maycee.padilla@deped.gov.ph, Department of Education- SDO Batangas Province, Batangas, Philippines. Assessment of teaching english language skills: Input to digitized activities for campus journalism advisers. *International Multidisciplinary Research Journal*, 4(4), January 2023.

[5] Guide to app architecture. https://developer.android.com/topic/architecture. [Online; accessed 10-02-23].

[6] Ronald T Azuma. A survey of augmented reality. *Presence: teleoperators & virtual environments*, 6(4):355–385, 1997.

[7] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *IEEE computer graphics and applications*, 21(6):34–47, 2001.

[8] Alcínia Z Sampaio, Miguel M Ferreira, Daniel P Rosário, and Octávio P Martins. 3d and vr models in civil engineering education: Construction, rehabilitation and maintenance. *Automation in construction*, 19(7):819–828, 2010.

[9] Jad Chalhoub and Steven K Ayer. Using mixed reality for electrical construction design communication. *Automation in construction*, 86:1–10, 2018.

[10] Sanika Doolani, Callen Wessels, Varun Kanal, Christos Sevastopoulos, Ashish Jaiswal, Harish Nambiappan, and Fillia Makedon. A review of extended reality (xr) technologies for manufacturing training. *Technologies*, 8(4):77, 2020.

[11] Beyond AR vs. VR: What is the Difference between AR vs. MR vs. VR vs. XR? https://www.interaction-design.org/literature/article/beyond-ar-vs-vr-what-is-the-difference-between-ar-vs-mr-vs-vr-vs-xr. [Online; accessed 18-03-23].

[12] Alexandru Predescu, Mariana Mocanu, and Ciprian Lupu. ARMAX: A Mobile Geospatial Augmented Reality Platform for Serious Gaming. In *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 313–318, Cluj-Napoca, Romania, September 2019. IEEE.

[13] Cristina Portalés, José Luis Lerma, and Santiago Navarro. Augmented reality and photogrammetry: A synergy to visualize physical and virtual city environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(1):134–142, January 2010.

[14] Samaneh Sanaeipoor and Khashayar Hojjati Emami. Smart City: Exploring the Role of Augmented Reality in Placemaking. In *2020 4th International Conference on Smart City, Internet of Things and Applications (SCIOT)*, pages 91–98, Mashhad, Iran, September 2020. IEEE.

[15] Fabio Latino, Vasilis Naserentin, Erik Öhrn, Zhao Shengdong, Morten Fjeld, Liane Thuvander, and Anders Logg. Virtual City@Chalmers: Creating a prototype for a collaborative early stage urban planning AR application.

[16] Arnis Cirulis and Kristaps Brigis Brigmanis. 3D Outdoor Augmented Reality for Architecture and Urban Planning. *Procedia Computer Science*, 25:71–79, 2013.

[17] Mohamed Zahlan Abdul Muthalif, Davood Shojaei, and Kourosh Khoshelham. A review of augmented reality visualization methods for subsurface utilities. *Advanced Engineering Informatics*, 51:101498, January 2022.

[18] Chenliang Wang, Kejia Huang, and Wenjiao Shi. An Accurate and Efficient Quaternion-Based Visualization Approach to 2D/3D Vector Data for the Mobile Augmented Reality Map. *IJGI*, 11(7):383, July 2022.

[19] Julian Keil, Dennis Edler, Thomas Schmitt, and Frank Dickmann. Creating Immersive Virtual Environments Based on Open Geospatial Data and Game Engines. *KN J. Cartogr. Geogr. Inf.*, 71(1):53–65, March 2021.

[20] Raghav Sood. *Pro Android Augmented Reality*. Apress, 2012.

[21] George Plakas, ST Ponis, K Agalianos, E Aretoulaki, and SP Gayialis. Augmented reality in manufacturing and logistics: Lessons learnt from a real-life industrial application. *Procedia Manufacturing*, 51:1629–1635, 2020.

[22] Daniel Broschart and Peter Zeile. Architecture: augmented reality in architecture and urban planning. *Peer reviewed proceedings of digital landscape architecture*, 2015:111, 2015.

[23] Hsin-Kai Wu, Silvia Wen-Yu Lee, Hsin-Yi Chang, and Jyh-Chong Liang. Current status, opportunities and challenges of augmented reality in education. *Computers & education*, 62:41–49, 2013.

[24] Nedal Sawan, Ahmed Eltweri, Caterina De Lucia, Luigi Pio Leonardo Cavaliere, Alessio Faccia, and Narcisa Roxana Moşteanu. Mixed and augmented reality applications in the sport industry. In *2020 2nd International Conference on E-Business and E-commerce Engineering*, pages 55–59, 2020.

[25] Jinhyuk Choi, Bongkyu Jang, and Gerard J Kim. Organizing and presenting geospatial tags in location-based augmented reality. *Personal and Ubiquitous Computing*, 15:641–647, 2011.

[26] G Bharath, Rupa Ch, M Karthik, Manish Chowdary, et al. Revelation of geospatial information using augmented reality. In *2021 Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 303–308. IEEE, 2021.

[27] Gerhard Schall, Stefanie Zollmann, and Gerhard Reitmayr. Smart vidente: advances in mobile augmented reality for interactive visualization of underground infrastructure. *Personal and ubiquitous computing*, 17:1533–1549, 2013.

[28] Thierry Badard. Geospatial service oriented architectures for mobile augmented reality. In *Proc. of the 1st International Workshop on Mobile Geospatial Augmented Reality*, pages 73–77. Citeseer, 2006.

[29] B St-Aubin, M Mostafavi, S Roche, and N Dedual. A 3d collaborative geospatial augmented reality system for urban design and planning purposes. In *Canadian Geomatics Conference and Symposium of Commission I, ISPRS, Convergence in Geomatics-Shaping Canada's Competitive Landscape*, volume 6, 2010.

[30] Gerhard Schall. *Mobile augmented reality for human scale interaction with geospatial models: The benefit for industrial applications.* Springer Science & Business Media, 2012.

[31] Bahman Jamali, Abolghasem Sadeghi-Niaraki, Reza Arasteh, et al. Application of geospatial analysis and augmented reality visualization in indoor advertising. *International Journal of Geography and Geology*, 4(1):11–23, 2015.

[32] Johan Kasperi, Malin Picha Edwardsson, and Mario Romero. Occlusion in outdoor augmented reality using geospatial building data. In *Proceedings of the 23rd ACM symposium on virtual reality software and technology*, pages 1–10, 2017.

[33] Jorge D Camba and Manuel Contero. From reality to augmented reality: Rapid strategies for developing marker-based ar content using image

capturing and authoring tools. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–6. IEEE, 2015.

[34] Kostas Anagnostou and Panagiotis Vlamos. Square ar: Using augmented reality for urban planning. In *2011 third international conference on games and virtual worlds for serious applications*, pages 128–131. IEEE, 2011.

[35] Yassir El Filali and Salah-ddine Krit. Augmented reality types and popular use cases. *International Journal of Engineering, Science and Mathematics*, 8(4):91–97, 2019.

[36] Ryan Bane and Tobias Hollerer. Interactive tools for virtual x-ray vision in mobile augmented reality. In *Third IEEE and ACM international symposium on mixed and augmented reality*, pages 231–239. IEEE, 2004.

[37] Benjamin Avery, Christian Sandor, and Bruce H Thomas. Improving spatial perception for augmented reality x-ray vision. In *2009 IEEE Virtual Reality Conference*, pages 79–82. IEEE, 2009.

[38] Alan Torres-Gómez and Walterio Mayol-Cuevas. Recognition and reconstruction of transparent objects for augmented reality. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 129–134. IEEE, 2014.

[39] Ming-Chang Wen and Shih-Chung Kang. Augmented reality and unmanned aerial vehicle assist in construction management. In *Computing in Civil and Building Engineering (2014)*, pages 1570–1577. 2014.

[40] Open Geospatial Consortium Standarts. https://www.ogc.org/standards/citygml. [Online; accessed 12-02-23].

[41] Gerhard Gröger and Lutz Plümer. Citygml–interoperable semantic 3d city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33, 2012.

[42] Tatjana Kutzner, Kanishk Chaturvedi, and Thomas H Kolbe. Citygml 3.0: New functions open up new applications. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 88(1):43–61, 2020.

[43] Android Distribution Chart. https://www.composables.co/tools/distribution-chart. [Online; accessed 11-02-23].

[44] What is CityJSON. https://www.cityjson.org/. [Online; accessed 02-02-23].

[45] Karin Starin. Combination of cityjson with postgresql, mongodb and graphql.

[46] Filip Biljecki and Ken Arroyo Ohori. Automatic Semantic-preserving Conversion Between OBJ and CityGML. *Eurographics Workshop on Urban Data Modelling and Visualisation*, page 6 pages, 2015. Artwork Size: 6 pages ISBN: 9783905674804 Publisher: The Eurographics Association.

[47] Christoph Blut, Timothy Blut, and Jörg Blankenbach. CityGML goes mobile: application of large 3D CityGML models on smartphones. *International Journal of Digital Earth*, 12(1):25–42, January 2019.

[48] Thomas H Kolbe, Gerhard Gröger, Lutz Plümer, et al. Citygml–interoperable access to 3d city models. *Geo-information for disaster management*, 49, 2005.

[49] Christoph Praschl and Oliver Krauss. Geo-Referenced Occlusion Models for Mixed Reality Applications using the Microsoft HoloLens:. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 113–122, Online Streaming, — Select a Country —, 2022. SCITEPRESS - Science and Technology Publications.

[50] Faraliyana Mohd Hanafi, Muhammad Imzan Hassan, Hanis Rashidan, and Alias Abdul Rahman. Web Visualization of 3D Strata Objects based on CityJSON and LADM. 2022. Publisher: FIG International Federation of Surveyors - Delft University of Technology.

[51] Android Application Fundamentals. https://developer.android.com/guide/components/fundamentals. [Online; accessed 19-02-23].

[52] Android is for everyone. https://www.android.com/everyone/enabling-opportunity/. [Online; accessed 19-02-23].

[53] Android's Kotlin-first approach. https://developer.android.com/kotlin/first. [Online; accessed 19-02-23].

[54] Android Studio Electric Eel 2022.1.1. https://developer.android.com/studio/releases. [Online; accessed 19-02-23].

[55] Android Studio projects overview. https://developer.android.com/studio/projects. [Online; accessed 28-03-23].

[56] Fundemental concepts of ARCore. https://developers.google.com/ar/develop/fundamentals. [Online; accessed 19-02-23].

[57] James R Vallino. Interactive augmented reality.

[58] ARCore Geospatial API Documentation. https://developers.google.com/ar/develop/geospatial . [Online; accessed 15-02-23].

[59] ARCore Geospatial location permissions. https://developers.google.com/ar/develop/java/geospatial/enable. [Online; accessed 19-02-23].

[60] Mobile Operating System Market Share Worldwide. https://gs.statcounter.com/os-market-share/mobile/worldwide. [Online; accessed 15-02-23].

[61] Christoph Blut and Jörg Blankenbach. Three-dimensional CityGML building models in mobile augmented reality: a smartphone-based pose tracking system. *International Journal of Digital Earth*, 14(1):32–51, January 2021.

[62] What is Android Studio / Download Android Studio. https://developer.android.com/studio. [Online; accessed 15-02-23].

[63] ARCore by Google. https://developers.google.com/ar. [Online; accessed 15-02-23].

[64] ARCore documentation. https://developers.google.com/ar/develop . [Online; accessed 15-02-23].

[65] Berlin 3D - Downloadportal. https://www.businesslocationcenter.de/berlin3d-downloadportal//export. [Online; accessed 05-02-23].

[66] FME | Data Integration Platform. https://www.safe.com/fme/. [Online; accessed 15-02-23].

[67] Material Design. https://m2.material.io/design. [Online; accessed 19-02-23].

[68] How to design a better bottom navbar (Tab bar)? https://uxplanet.org/how-to-design-a-better-bottom-navbar-5127c8b8102f. [Online; accessed 15-02-23].

[69] Figma Design Tool. https://www.figma.com/. [Online; accessed 19-02-23].

[70] How to migrate splash screens. https://developer.android.com/develop/ui/views/launch/splash-screen/migrate. [Online; accessed 19-02-23].

[71] Android Fragments. https://developer.android.com/reference/androidx/fragment/app/Fragment. [Online; accessed 28-03-23].

[72] Request location permissions. https://developer.android.com/training/location/permissions. [Online; accessed 19-02-23].

[73] ARCore Geospatial Codelab project. https://github.com/google-ar/arcore-android-sdk/tree/master/samples/geospatial$_java. [Online; accessed 10-02-23]$.

[74] ARCore Geospatial Sample. https://developers.google.com/ar/develop/java/geospatial/enable. [Online; accessed 10-02-23].

[75] Creating of custom 3D models. https://developers.google.com/ar/develop/c/augmented-faces/create-assets. [Online; accessed 10-02-23].

[76] Karin Jacquelien Staring. Suitability of MongoDB compared to other databases for the storage and querying of CityJSON using GraphQL.

[77] Unity Game Engine. https://unity.com/. [Online; accessed 10-02-23].

[78] Unreal Game Engine. https://www.unrealengine.com/en-US. [Online; accessed 10-02-23].