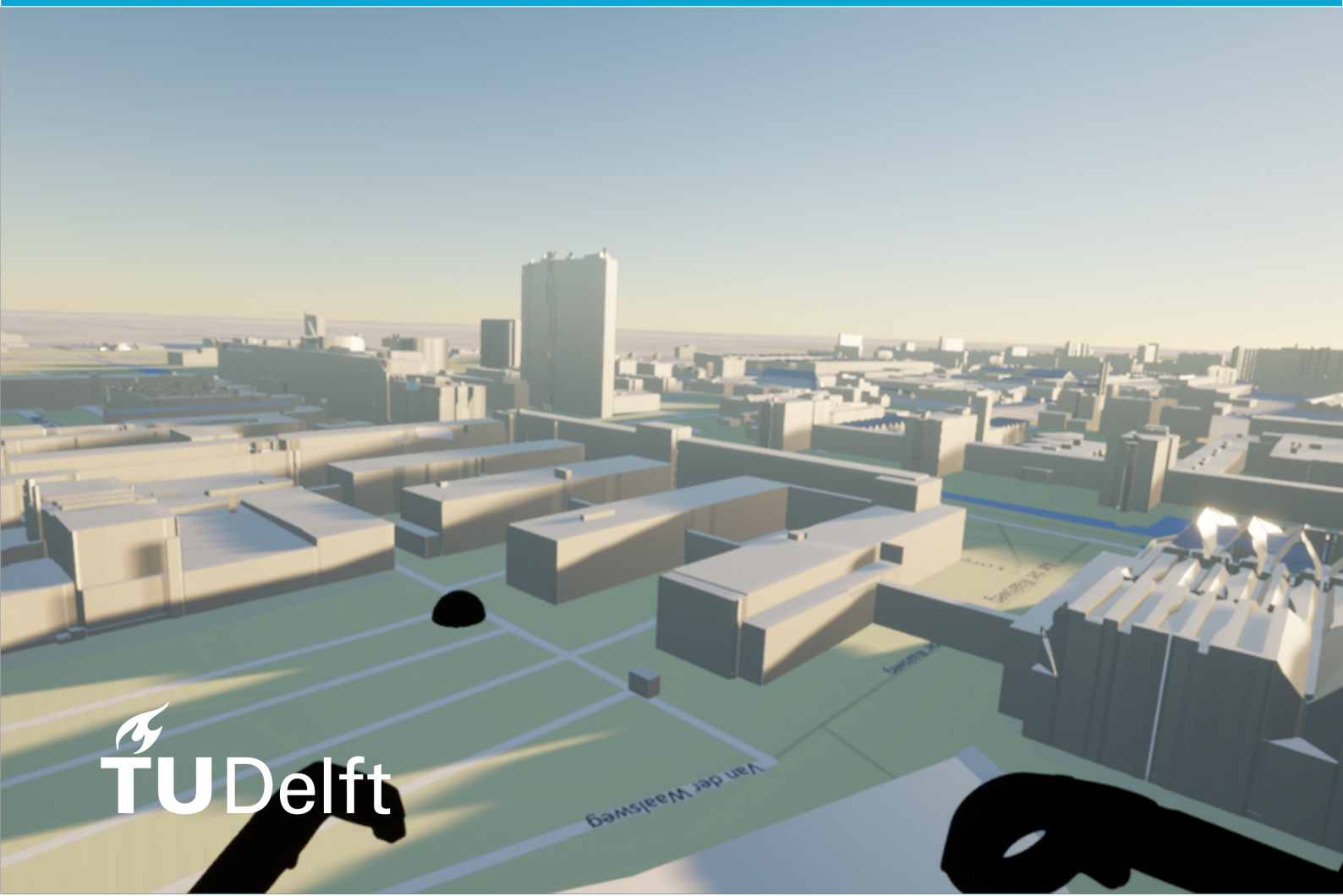
MSc thesis in Geomatics for the Built Environment

An OGC 3D tiling technique based user-interactive platform for digital twins in a virtual reality environment.

Pratyush Kumar
2022



TU Delft

AN OGC 3D TILING TECHNIQUE BASED USER-INTERACTIVE PLATFORM FOR DIGITAL TWINS IN A VIRTUAL REALITY ENVIRONMENT

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Geomatics for the Built Environment

by

Pratyush Kumar

June 2022

Pratyush Kumar: *An OGC 3D tiling technique based user-interactive platform for digital twins in a virtual reality environment* (2022)

The work in this thesis was made in the:

Geo-Database Management Centre
Chair of GIS Technology
Section Design Informatics
Architectural Engineering and Technology Department
Faculty of Architecture & the Built Environment
Delft University of Technology

Supervisors: Dr. Azarakhsh Rafiee
Dr. Martijn Meijers
Co-reader: Dr. Stefan van der Spek

ABSTRACT

There has been an ever increasing focus on the development of smart cities, both by big corporate in the field of Information Communication and Technology as well as respective governments responsible for these cities worldwide. This in combination with the influx of easily accessible data owing to cheaper and efficient sensors has risen to the concept of digital twinning of urban spaces and cities.

Urban digital twins require the city to be represented in 3-dimensional digitally, these datasets are oftentimes too large to be provided on as-is basis for most platforms. To tackle this problem of large 3D datasets, the OGC 3D tiling specifications were formed and accepted in 2019.

This thesis attempts to develop a neighbourhood designing platform which would enable users to design neighbourhoods in 3D while being able to experience the design in an immersive manner by utilizing virtual reality's capability to observe 3D assets in scales which are not possible using a physical model or a 2 dimensional neighbourhood plan, employing the OGC 3D tiling technique to understand the extent to which a combination of existing 3D datasets can be made with a user-interactive neighbourhood designing platform in a VR environment.

The results of the study reveal that using our methodology, it is possible to combine existing 3D datasets with user made neighbourhood design with their own 3D assets of any level of detail and furthermore, it is possible to utilize instanced tiling format to disseminate the neighbourhood design as a standard 3D tiled design.

ACKNOWLEDGEMENTS

I would like to extend my thanks to my mentors, Dr. Azarakhsh Rafiee and Dr. Martijn Meijers for enabling me to take up this research subject and for providing me with valueable feedback, guidance and questions along the way, throughout this thesis. I would also like to thank my parents, sister and my close friends, especially Theodoros, Giorgos, Giannis, Ajit, Iris, Avanika, Oliver, Swareena, Parneet and Krish for keeping my motivation up when it was needed and for being supportive to me. Lastly, I would like to thank the student assistants at the VR lab in the Faculty of Architecture and Mr. Hans Hoogenboom for helping me out with the application development in Unreal Engine, when I could not figure out how things could be done, and for providing me with the opportunity to work at the VR lab with the VR equipments.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Algorithms

ACRONYMS

1 INTRODUCTION

1.1 MOTIVATION

In the last decade, we have seen a huge influx of the availability of data and its usage to find correlations between subjects that had never before been thought to be linked. Today, information technology has become one of the main component driving our economies and leading to modern development. We have come across the advent of faster and better processing power of computers, Internet of Things, and Big Data, across all sectors and fields of studies.

With the onset of this digital-revolution and the spread of cheaper access to internet all across the world, there has been a significant boom in the number of Data-Driven-City and Smart City projects [Ivanov et al., 2020] undertaken for various cities throughout the world. This onset of focus on smart cities has also brought about a lot of digitization of urban datasets, capturing of these datasets using sensors placed across the city as well as visualization of the raw or analysed data onto visualization platforms. While in the field of manufacturing and space applications, simulations as well as digital twinning of complex processes have been used for better analysis and decision-making for a long time with the first instance being traceable to the NASA Apollo 13 mission [Cureton and Dunn, 2021; Ferguson, 2020; GlobalData Thematic Research, 2020], recent advances in technology as well as the conception of smart cities have enabled the digital twinning of smaller process to be scaled up to a larger, urban scale.

In the field of urban planning, three-dimensional (3D) models have been in use for some years to represent the physical form of the city, by providing a visualization of the physical elements of the city such as its buildings, public spaces and transportation networks. They have also been employed and developed for use cases of disaster management, energy demand analysis, infrastructure planning, urban heat island analysis, etc [Biljecki et al., 2015]

An urban digital twin is a virtual representation of the real city, with all the real-time datasets being analysed and reflected at one single platform [Shahat et al., 2021]. This platform can then be used by different professionals involved in the process of urban planning to take effective and well-informed decisions for the cities [Shahat et al., 2021]. While in recent years, such platforms have been developed, they have mostly been used as a tool for visualization of cities in 3D, however, in addition to usage of such platforms for the purpose of visualization, it could be used to implement or develop a scenario analysis platform. This scenario analysis platform could then be employed in different use-cases of urbanism, one of them being urban neighbourhood design. This scenario analysis can be done not only in the early design phase where alternative design scenarios can be tested but also for post-construction period where the platform could be used for validation and calibration of the design and models.

In the case of urban digital twins, the 3D city models being used to depict the city, can, in itself be relatively large for the user to load over the network without waiting long for it, for example, the CityGML file for the city of New York has a size of 590 MB [Ledoux, 2019], while there are researches in the field of 3D city models to reduce their size, like the OGC standard of CityJSON [Ledoux et al., 2019], the file size still remains quite large to be downloaded on the client side. Countering this problem of large city models, in 2019, OGC accepted and published, a 3D tile

standard for streaming and rendering massive 3D geospatial data, this 3D tiling technique would be used for the visualization of the 3D city models on the platform [Open Geospatial Consortium, 2019]. The tiling standard ensures that large datasets have the capability for streaming them as 3D tiles, thus the client device only needs to render a small part of the visible frustum of the city model and not the entirety of it, only tiles which are visible are loaded. Moreover, in case of neighbourhood design it should be possible to export the designed area as a 3D tile, to ensure that the data format is standardized.

When studying urban planning and design as a budding planner, I realized that most of the spatial plans that were being designed were two-dimensional (2D), meaning that they were designed with only the plan-view. However, with more and more datasets being made available in 3D, and with the possibility to easily transfer datasets in 3D, it is high time that urban planning be done in 3D as well. Moreover, this would help ensure that we start designing our plans for not only 2D cadasters but also for 3D cadasters. While 3D models are created to facilitate the plan-view based designing process, the models are often scaled down and thus do not provide the same immersive experience one could get by visualising the 3D model in virtual reality (VR). This study aims at developing a neighbourhood designing platform leveraging the OGC 3D tiling technique in a VR based environment to ensure an immersive user experience.

1.2 RESEARCH QUESTION

1.2.1 Scope and Objectives

The scenario analysis platform, by its virtue, should be able to run user-interactive simulations on the city model and make the visualizations possible. Some of these simulations could be: traffic simulations in case of change of land use, or, neighbourhood development simulation, with the possibility for the user to change and define the floor area ratio, types of building, circulation pattern, ground coverage, urban population density, housing density, etc. This study will focus on the development of a user interactive platform for neighbourhood design with possibility to visualize and use the platform using VR.

The platform would be developed in a game engine, Unreal Engine, which would also allow us to make a more immersive experience for the end-user (by using VR). The VR enabled platform is of use case in urbanism, especially so in the case of public participation or decision-making, by enabling the end users to experience the proposed design using virtual visualizations.

This research could be taken into many directions, however, it will solely focus on the development of the platform, this would not be a research on finding the most optimized type of tiling technique that could be incorporated, or a research on how effective is VR platform for the purpose of urban designing as compared to traditional CAD, or a research on automated designing of an urban neighbourhood. These topics can be researches on their own and can be considered as future scope of the study as detailed out in Section 6.3.

1.2.2 Research questions

The main research question for this thesis is: **"To what extent can we make a combination of existing 3D city datasets with a user interactive neighbourhood designing platform in a virtual reality environment?"** In order to attain the objectives specified in the above subsection, the following research sub-questions need to be addressed:

- Can the 3D tiling technique be utilized for the use case of urban neighbourhood development?
- In the use case of urban neighbourhood design, how can dynamic user selected designs be effectively visualized?
- How can user-designed neighbourhoods be dynamically disseminated in the form of 3D tiling to incorporate it with the rest of the contextual 3D tiles?

1.3 THESIS OVERVIEW

In this thesis, Chapter 2 describes all the theoretical background required to answer the aforementioned research questions. Chapter 3 details out the steps needed to be carried out to answer the research questions established in Section 1.2.2. Following the methodology in Chapter 4, the implementation details of how the methodology was carried out is discussed, including the datasets used and data preparation. In Chapter 5, we discuss our findings from this study and discuss upon the results gathered. In Chapter 6, we attempt to answer the research questions of this study and followed by a discussion on the limitations and the future scope of this study.

2 THEORETICAL BACKGROUND & RELATED WORK

In this chapter we will discuss the required theoretical background needed to answer our research questions established in Chapter 1. The areas of interest for this study pertain to 3D city models, 3D tiling and its techniques and Game engine and their usage in urban design and visualizations, especially for the case of VR visualization.

2.1 3D CITY MODELS

City models are representations of the urban environment and the urban entities of the real world geographic locations [Billen et al., 2014]. A 3D city model is a city model represented with the three-dimensional geometries of the urban fabric which include the buildings, landscaping elements and other objects [Biljecki et al., 2015].

In the past decades, there have been many use cases of 3D city model, but mainly focusing on visualization of the urban environment [Chen, 2011; Kolbe, 2009]. With the improvement of GIS standards and simulation models, the use cases of 3D city models have also increased. This rise in the number of usage can be seen ranging from the fields of environmental simulation to urban planning, which deal with the problems faced in the urban environment like traffic network simulations, disaster management etc [Biljecki et al., 2015].

In the field of urban planning, 3D city models could help planners to visualize and simulate the urban environment to their understanding, and to see how things would differ if a certain plan were to be implemented versus another plan. This enables the possibility to interact with design choices [Chen, 2011] and then do scenario analyses based on the different simulations produced from the large urban datasets [Chen, 2011; Stojanović et al., 2014]. In some studies such as those by Al-Douri [2006]; Onyimbi et al. [2017], 3D city models are also shown to be stepping stones towards better and more inclusive public participation as it helps the common citizen to better comprehend the changes that would be brought about by certain policy changes or design changes to the city [Chen, 2011].

2.2 URBAN DIGITAL TWINS

The term Digital Twin (DT) was coined by Professor Grieves in 2003 in his lecture at the University of Michigan, however, the idea of twinning of physical processes so as not only to simulate pre-existing conditions but also to adapt the simulation to rapidly changing conditions digitally had already existed and could be traced back to as early as NASA's Apollo 13, where computational simulations could be run in real time to make strategic decisions following mid-flight failure of components [Ferguson, 2020]. These digital simulations enable the user to assess performance of certain tasks using "what-if?" scenarios, improvements can be made to the product in a virtual environment and further tests can be conducted on the same, virtually [Ayres, 2012].

It is now possible to scale up this idea of digital twinning of processes to a scale of entire cities, to establish entire nations' cities digitally [Rogers, 2019]. Although this has been around for some years, it is still in a nascent stage of development.

City digital twin or Urban digital twins can be said to be an evolution of Internet of Things and smart city concepts, emerging from a mixture of geographical information system (GIS) and building information model (BIM) [de Laat and van Berlo, 2011]. This fusion of GIS and BIM has many benefits where complex urban management is facilitated by leveraging information modelling of BIM and networking of GIS. These city digital twins can be used to for testing of scenarios, improved decision-making, operation, etc. While BIM and GIS have been integrated for some time, we now also have large scale 3D city models in higher Level of Detail (LoD) such as LoD2. For the case of this study, we will use such a 3D city model and not BIM datasets, since the focus is on a city scale perspective and not on individual buildings.

A very exhaustive list of possible analysis and visualizations with the help of 3D city models can be found in the research done by Biljecki et al. [2015], listing a wide range of possible use cases of 3D city models, mainly dividing them into two main use cases:

1. Non-Visualization based use cases such as solar irradiation estimation, energy demand estimation, floor space determination, build typology classification
2. Visualization based use cases, such as: visibility analysis, 3D cadastre, visualization for navigation, facility management, emergency response management, lighting and shadow simulation, flood vulnerability estimation, forest management, population estimation, etc

While these use cases are stated to be for the application of 3D city models, an urban digital twin will essentially use a 3D city model in the contextual background and display other datasets on top of it, which could be both static, pre-defined analysed data or real time Application Programming Interface (API) based datasets.

2.3 3D TILES

2.3.1 OGC 3D tiles

2D tiles have been in use for the purpose of displaying 2D datasets on a map at varying scale. Similar to 2D tiles, 3D tiles is an open specification developed for "sharing, visualizing, fusing, interacting with, and analysing massive heterogeneous 3D geospatial content across desktop, web, and mobile applications" [Cesium GS Inc, 2020]. 3D tiles are built on an open standard, GL Transmission Format (glTF), which is an open source, royalty-free specification for efficient transmission, streaming and rendering of 3D assets by engines and applications. The 3D assets are minimized by glTF along with the runtime processing required for unpacking them. glTF also enables interoperable use of 3D content across different platforms [Khronos Group, 2017].

OGC 3D tiling allows for various tile formats such as: Batched 3D models, instanced 3D models, point clouds, composite tiles [Cesium GS Inc, 2020]. The details of each type of tiling technique is described in the table Table 2.1.

In Cesium, the 3D tiles are rendered according to a scheduling strategy, which are defined by various scheduling parameters, the most important ones of them being the screen spatial error (SSE). The tile in 3D tile is dependent on the factors of its geometric error and the bounding volume. When rendering the tile, Cesium first checks for the topological relationship of the intersection of the bounding volume with the view frustum. If the bounding volume is within, i.e. if the intersection yields true, then Cesium checks the geometric error for the tile, if it is within the pre-specified geometric error as specified in the tileset, then the tile is rendered.

3D City models have become more and more in use in various urban projects. Table 2.2 lists some working cases in the Netherlands and elsewhere which have made a 3D city model visualization platform.

Format	Extension	Usage
Batched 3D models	.b3dm	Heterogeneous 3D models. E.g. textured terrain and surfaces, 3D building exteriors and interiors, massive models.
Instanced 3D models	.i3dm	3D model instances. E.g. trees, windmills, bolts.
Point Cloud tiles	.pnts	Massive number of points.
Composite tiles	.cmpt	Concatenate tiles of different formats into one tile.

Table 2.1: 3D tiling formats as per OGC specification [Open Geospatial Consortium, 2019]

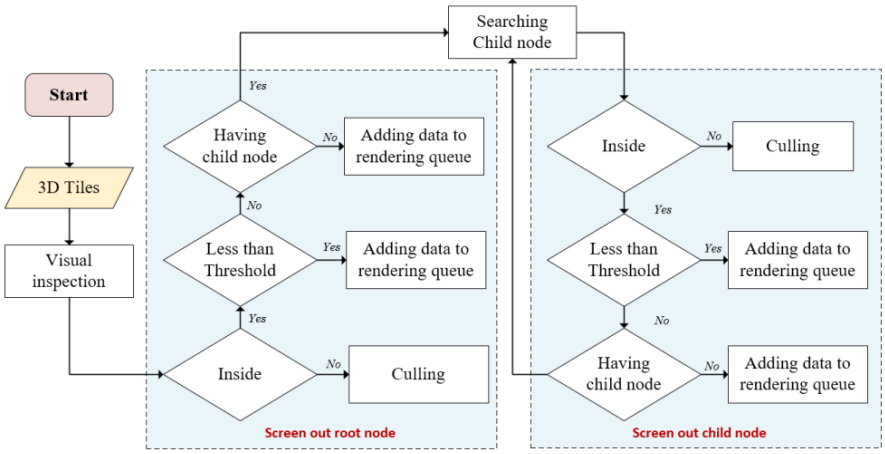


Figure 2.1: Cesium Scheduling strategy from ([Cesium GS Inc, 2020])

Project	Area	Details
3dbag.nl	The Netherlands	A team at the 3D geoinformation research group of TU Delft has developed a countrywide, up-to-date 3D registration of building which uses tile based viewing for its web-viewer [BAG, 2021].
3d.amsterdam.nl	Amsterdam	The municipality of Amsterdam has developed a 3D model of the city of Amsterdam which allows for user interaction with the web-viewer. The dataset used is from 3D Basisregistratie Adressen en Gebouwen (BAG), and the platform is developed using Unity [Amsterdam, 2021].
3drotterdam.nl	Rotterdam	Similar to 3D Amsterdam, maintained by Municipality of Rotterdam [Rotterdam, 2021]
3d.utrecht.nl	Utrecht	Similar to 3D Amsterdam, maintained by Municipality of Utrecht [Utrecht, 2021]
Gothenburg City	Gothenburg city, Sweden	Digital twin of the second largest city of Sweden made by using FME, ESRI City Engine and Unreal Engine
Virtual Singapore	Singapore	Government sponsered city digital twin for Singapore, made with an intent to function as a collaborative platform to be used by public, private and research sectors.

Table 2.2: Previously implemented projects incorporating 3D city models.

Some other use cases for the usage of 3d tiles and their users are summarised in Table 2.3

User	Use Case
maxxar Precision 3D	accurate 3d surface model for digital twins and other use cases
Swiss Topo	provides 3D tiles based visualization of Swiss cities
benteley context capture	generate Cesium based 3d tiles models
cybercity3d	creates digital twins and uses 3d tiling for visualisation
virtual city systems	used 3d tililng to visualize and enable planning operations
CityZenith	using urban digital twin to reduce carbon emission
CSIRO Data61	3d tiles based digital twin for visualization and decision making support using VR
Georocket	stores 3d geospatial files and serves them as 3d tiles for visualisation
Gamesim Conform	3D Geospatial Software for fusing, visualizing, editing, and exporting beautiful 3D environments for urban planning, games, and simulations.
SiteSee	uses 3d tiles for digital twins of infrastructure
Virtual GIS	Visualize infrastructure deployed in 3d using 3d tiles
LOPoCS Pointcloud viewer	uses 3d tiles to visualize point clouds
GeoPipe	Earth-wide digital twin built by AI, based on unity
3D Digital Territory Lab	3d tiles used for integrating various city wide datasets in form of DEM, point cloud, ortho imagery, WMS services etc
Çeşme 3D City Model	turkey 3d digital twin using 3d tiles for visualization

Table 2.3: Users and their use cases of OGC 3D tiles

2.3.2 3D tiling platforms

The OGC 3D tiling was proposed to OGC by Cesium. Cesium not only supports the viewing of the 3D tiles but also the formation and web-based hosting of these tiles using Cesium ion, which takes 3D datasets and then tiles them on the web and also serves the tiles using a REST API. Cesium has also collaborated with Safe FME to incorporate a Cesium plugin in FME which can be used to create batched tile models. In addition to these platforms, the Geodan pg2b3dm project from Geodan focusses at the conversion of PostgreSQL city datasets (PostGIS) to b3dm batched tilesets for use in Cesium or in MapboxJS [Geodan, 2022]. There are numerous other open-source or proprietary software programs which enable us to create 3D tiles from different 3D file formats, like obj23d-tiles [Jiang, 2019] or geopipe's gltf2glb tool [Geopipe, 2022].

2.3.3 Cesium

Cesium supports the loading and display of 3D tiles. In case of its JavaScript based platform CesiumJS, all the formats described in Table 2.1 are natively supported and can be served into the JavaScript application using a locally hosted server which

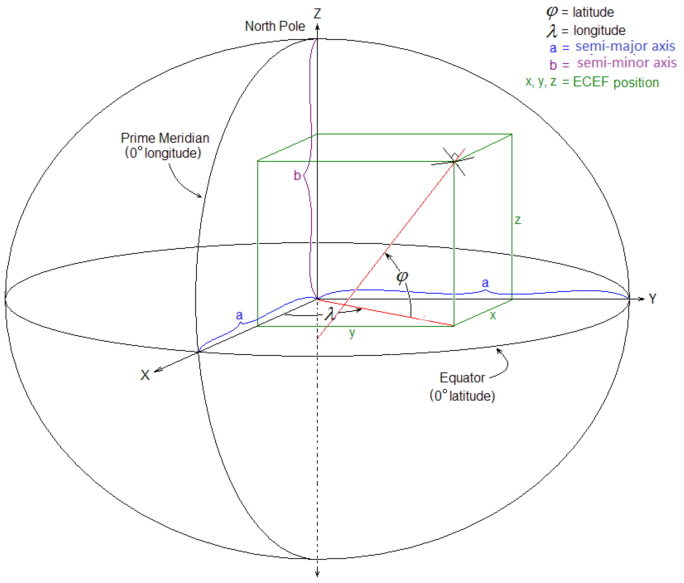


Figure 2.2: ECEF and East North Up with respect to Earth (from [Wikipedia, 2021])

hosts the tiles. Cesium however uses an Earth-Centered, Earth-Fixed (ECEF) coordinate system in which the centre of mass of the reference ellipsoid of Earth is taken as origin. The 3D tiles served to Cesium must be in the EPSG:4978 coordinate system for the objects to be rendered at the correct location. In this coordinate system the ellipsoid used for representing earth is the same as WGS84 ellipsoid, the Geocentric X-axis passes through the prime meridian, the Y axis through a right-handed orthogonal system by a plane 90° east of the X-axis and its intersection with the equator, and the Z axis passes through the North Pole [Dana, 1995].

2.4 GAME ENGINE

Over the past years, game engines have found usage in the simulation of different environments and ecosystems, such as planning of fire escape routes in an indoor model of a building, or simulation of the road traffic network of a city [Besuievsky and Patow, 2013; FU et al., 2021; Rüppel and Schatz, 2011], than just creation of cross-platform games. One such application has been found in the case of urban analysis; case studies such as [ESRI, 2019; Indraprastha and Shinozaki, 2009] show the usage of game engines in visualization of urban design projects in which the user can interact with the model, however some analysis can also be conducted on urban data using game engines, such as finding the historical path using network analysis [Vletter, 2019] or calculation of rooftop solar potential [Buyuksalih et al., 2017].

The usage of 3D city models in a game engine brings about possibilities for visual and non-visual use cases [Biljecki et al., 2015] such as: energy demand application, rooftop solar potential estimation, visibility analysis, dynamic scenario based urban design, urban heat island analysis, etc. As the user can interact with these models in a game engine, from a visualization perspective, it also enables the urban planners to view the simulated changes, with better spatial awareness and understanding to take better design decisions [Indraprastha and Shinozaki, 2009].

The game engine enables for the study to incorporate some of its powerful features such as game physics, which make simulation of urban environments look realistic as well as better rendering capabilities for large 3D datasets, thus proving to be highly advantageous to the use-case of this research. There are multiple game engines available for producing various games, some of them being: Unity, Unreal

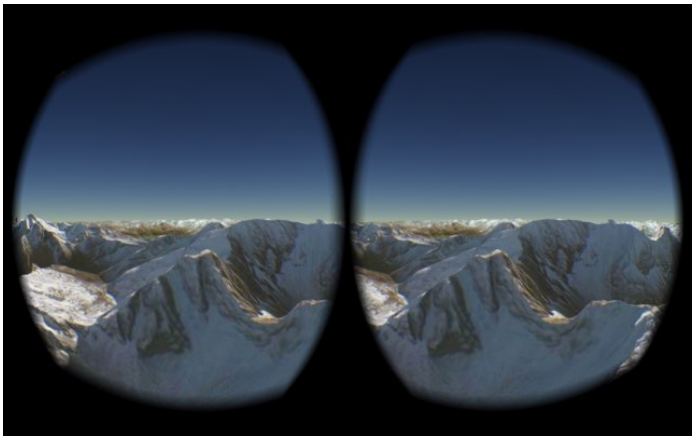


Figure 2.3: VR stimulus to both eyes (from [CSIRO, 2016])

Engine, Amazon Lumberyard, CryEngine, Godot, Open3D Engine, etc. For the purpose of this study, the game engine, Unreal Engine is chosen in order to leverage the Cesium for Unreal Plugin available with Unreal Engine, enabling us to render 3D tiles within a game engine using Cesium. In the case of the game engine: Unreal Engine 4.27, the physics engine used is known as PhysX [Epic Games, 2021a]. This physics engine handles the effects of gravity, collision, destruction of objects, the 2D and 3D coordinate systems, and even tracing of light rays, Ray Tracing to render life-like images at a high frame rate by using the 3D objects and its associated material textures as well as the normals on the surface for scattering of light rays. The usage of a game engine also eases the process of handling user interaction with the application and enables us to create a virtual reality based application with ease [Epic Games, 2021b]. From the book by Lavalle [2020], we see how VR based applications are formed by taking two camera angles within the application looking at the same scene by they are roughly apart at the same distance as human eyes, thus enabling a stereoscopic vision to be formed. The game engine can handle multiple camera angles at the same runtime in the application, thus easing out the process of employing a VR based visualization in the application. This includes the formation of the stereoscopic view using 2 cameras and then post-processing the obtained view to fit the human optical perception. Figure 2.3 shows a preview of what the VR stimulus to both eyes looks like.

2.5 GEO-DESIGN PLATFORMS

While this study focusses on development of a neighbourhood designing platform, there are other geo-designing platforms available as well which do not employ 3D tiling technique for streaming 3D datasets. Two such platforms are discussed below: Tygron geo-design platform and ESRI City Engine. Tygron Engine is a software platform where multiple GIS analyses can be performed in 3D [Tygron, 2022], such as: hydrological analysis, rainfall flow analysis, watershed analysis, etc. It also enables the users to make parametric designs and includes 3D neighbourhood development capabilities, however, Tygron platform does not employ the OGC 3D tiling technique within its framework. ESRI CityEngine is another such geo-design platform owned by ESRI, which has capabilities for 3D modelling of entire cities [ESRI]. Moreover, it can also be linked to other GIS datasets to run spatial analyses or simulations, however, ESRI CityEngine does not use OGC 3D tiles natively as well.

3 METHODOLOGY

In this chapter, we discuss the proposed workflows that need to be taken in order to answer the research questions of this study as established in Section 1.2 of Chapter 1. We start with the overview of the methodology, which is broken down into several steps. To answer the question of *To what extent can we make a combination of existing 3D model with a user interactive platform in VR environment?*, we need to be able to load a 3D tiled model of the city and then be able to design a neighbourhood after being able to remove the objects within the AoI. While designing the neighbourhood, the user also needs to be prompted about the proposed land use of the area, so the design is following the land use plan proposed by the planning authority of the AoI. This would answer our research question pertaining to the visualization of neighbourhood design.

Furthermore, the user needs to be able to load their own 3D models onto the AoI to place them at various locations for the neighbourhood designing process to be achieved. Once this step is completed, we can then visualize the contextual 3d city model and the user made design together in the same space, followed by a VR based visualization, thus, enabling us to answer our research sub-question on the dissemination of user-made design with the contextual city model. Figure 3.1 summarizes the software flow model for the platform developed in this study.

3.1 LOADING CONTEXTUAL 3D CITY MODEL

Though 3D city models are available in different file formats, such as CityJSON and CityGML, for the purpose of this research, we need to be able to load the entire model as 3D tiles and then load it into our gaming engine for visualization. For doing so, there are two proposed workflows:

1. The 3D tiles are hosted locally on the device and then be used in the gaming engine, or
2. The tiles are hosted on a server and then called into the gaming engine using an API and its key

The tiling of the dataset is achieved using the open-source tool pg2b3dm from Geodan; this tool converts a PostgreSQL database to a b3dm tileset and is based on the working of py3dtiles Algorithm 3.1 walks through the code implemented by geodan pg2b3dm tool to create b3dm tiles from PostgreSQL database.

As an alternative to the pg3b3dm tool for creation of the 3D tilesets, the Cesium 3D tiles writer from Safe FME could also be used, to create a subset of the data in the required coordinate reference system to work on, which can then later be expanded to the entirety of the country 3D data.

For hosting the 3D tiles through a web server, we can also use Cesium ion, which can accept a multitude of data-types and then tile them on the server side, which can then be loaded using an API key into the gaming engine.

3D tiles employ Hierarchical level of detail, meaning that the tiles which are further away from the camera are not rendered. If the dataset is provided at multiple levels of detail then the tiles which are further away are rendered at a lower level of detail as compared to tiles which are in immediate vision. For this study, we will utilize a single LoD dataset for the contextual 3D tiles since both the process of

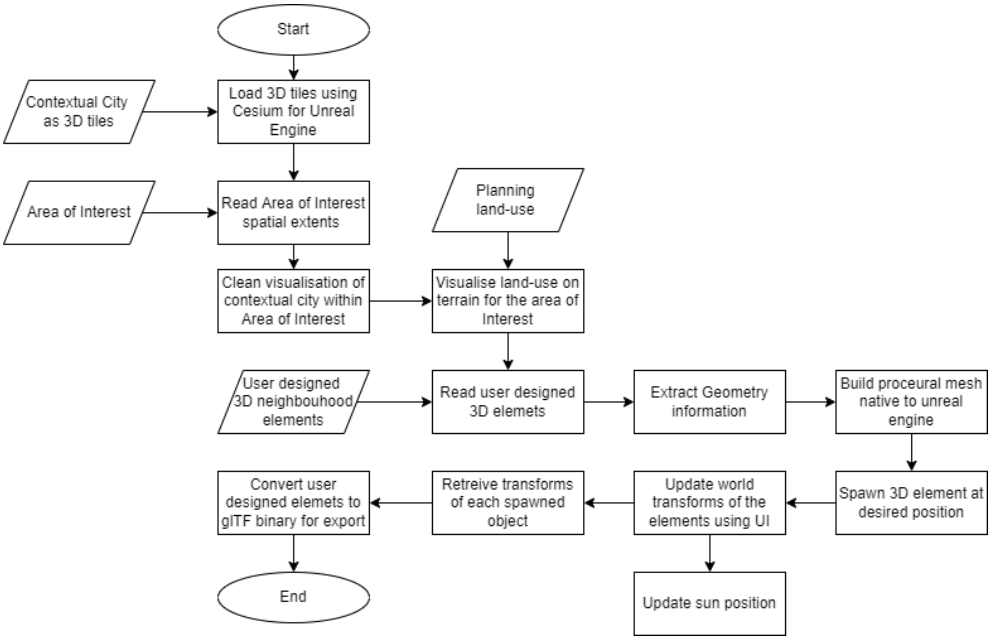


Figure 3.1: Flowchart depicting the flow model of the developed platform

Algorithm 3.1: Pseudocode for pg2b3dm tool for making batched tilesets

Input: user command *o* with: postgis database *db*, port *p*, User *u*, tile extent *tileExtent*
Output: a tileset.json file and its .b3dm tiles
Data: PostgreSQL database

```
while program ends or runtime error do
    conn = trusted connect to PostgreSQL database with command:
     "Host=o.Host;Username=o.User;Database=o.Database;Port=o.Port" ;
    if conn not trusted then
        pwd ← ask user for password ;
        conn = trusted connect to PostgreSQL database with command:
         "Host=o.Host;Username=o.User;Database=o.db;Port=o.p;password=pwd"
         ;
    create output directory ;
    bbox3d ← get bbox3d from conn;
    translation ← getCenter(bbox3d) ;
    bboxAll ← translate bbox3d to translation and rotate along X axis ;
    box ← get bounds of bboxAll for writing in json ;
    tiles ← get list of tiles according to o.tileExtent from bbox3d ;
    set bounding volume of each tile ;
    tileset.json ← write json for each tile ;
    /* writing tiles                                                    */
    foreach tile t in tiles do
        geom ← get geometry inside tile from PostgreSQL;
        triangles ← get triangles of geom;
        attributes ← get attributes of geom;
        create and write .b3dm tile;
```

creation of 3D tiles from 3D BAG dataset limits us from utilizing multiple LoD in a single tileset. However, since the cesium for unreal engine tool is employed for rendering the 3D tiles, it natively reduces the level of detail of the tiles rendered far away from the view frustum.

3.2 VISUALIZING 3D TILE-SET IN AREA OF INTEREST

After the loading of the contextual 3D city model, it is required for the user to delineate an AoI within which they plan to design a neighbourhood. This can be achieved by ingesting an AoI made at run-time by the user, or by ingesting an AoI which had been made by the user (in form of a geojson file). This delineated boundary is then topologically intersected with the contextual 3D city model and the buildings which happen to fall within the intersection would by stopped from rendering, thus emptying out the AoI of all pre existing structures.

Alternatively, Cesium for Unreal Engine plugin gives us the option to use a Cartographic Polygon as Raster Overlay component which can be attached to our 3D contextual city and thus enabling us to remove the visualizations within the AoI. The working of the Cartographic Polygon as Raster Overlay is elaborated in Algorithm 3.2

Algorithm 3.2: Pseudocode for the workings of Cartographic Polygon as Raster Overlay

Input: key press event k, 3d tileset *tiles3D*
if k **then**
    poly ← convert user input locations to Unreal polygon type;
    Cesium Cartographic Polygon *cartoPoly* ← attach *poly* ;
    cartoPoly ← attach *CesiumGlobeAnchorComponent* ;
    tiles3D ← attach *cartoPoly* as *CesiumPolygonRasterOverlayComponent* ;
    tiles3D ← set *materialLayerKey* as *clipping* ;

3.3 PLANNING BASE-LAYER

For neighbourhood design, the design needs to follow the proposed Land Use plan of the area it is being designed for. This proposed layout plan needs to be displayed once the cartographic polygon component is attached to the tileset to stop the visualization of the objects within the 3D city tileset. This layout plan is displayed on the background as a Web Map Service (WMS) served directly from whatever source that hosts the WMS from the planning department of the city, in the case of this study, from Ruimtelijke Plannen, or it can be served as hosted 3d tilesets, tiled using the same procedure as Section 3.2

Algorithm 3.3: Pseudocode for Cesium WMS overlay

Input: keypress event k, cesium terrain tiles t, game viewport v
if k **then**
    get view frustum of v;
    bounds ← extract view bounds from v;
    wms ← call WMS for *bounds* and get tiles ;
    t ← attach *wms* as Cesium WMS Raster Overlay;
    set material layer key to *Overlay*1

After the buildings within the AoI have been removed from the visualization, the overlay component is added to the 3D terrain tiles, which display the terrain below the buildings. This procedure is detailed in the Algorithm 3.3.

3.4 ENABLING USER DESIGN

To answer the research sub-question pertaining to dynamically visualizing and disseminating user-made design as 3D tiles, we need to be able to build a neighbourhood design by using user inputs and user interaction. This is achieved by enabling the user to place the elements of the neighbourhood design such as houses/buildings, trees, street furniture and other such elements into the scene once the contextual city is loaded and the buildings inside the AoI are removed from the visualization and the proposed land use layer is displayed in the background. These 3D objects could be provided in the form of a library, however since every study area is unique in its characteristic, the user needs to be given the power to use their own 3D elements and place them in the AoI. In the following subsections, the methodology proposed for enabling user to design their own neighbourhood is discussed. The steps followed are summarized as follows:

- Place user-made 3D neighbourhood design elements in the AoI.
- Alter the position and rotation of the placed objects.
- Enable visualization of different shadows for different times of the day for the user-placed 3D objects.

3.4.1 Loading model for user input from library

3D models used in neighbourhood design are comprised of vertices and faces formed by the connection of these vertices. These vertices, triangles (faces) and their texture, in combination form the 3D object which is rendered by the application. For the purpose of this research, the possibilities for loading in a 3D object which is then used for designing the neighbourhood can be achieved in the following ways:

- The application provides the user with a library of 3D objects which the user can drag and drop into the AoI
- The application lets the user input their own 3D objects which can then be placed within the AoI
- The application provides with a ready-made library of objects but also enables users to ingest their own models and 3D objects which can then be placed within the AoI using the user interface provided by the application.

In order to ingest a 3D object that has been made by the user, the vertices and triangles forming the faces of the 3D object need to be exposed to the platform developed in this study, which would then use these to render the object at runtime. The implementation of this runtime loading of user model is discussed in Section 4.5.1.

3.4.2 Enable editing 3D object position, orientation and scale

The models that are spawned by the user into the AoI are not spawned at the right orientation by default, moreover, the user might want to change the location of the spawned object in terms of it location in the world, its rotation or its scale. The application being developed needs to update the location and orientation of the spawned 3D object as per the user needs. In case of the application development environment of this study, Unreal Engine, this is achieved by updating the transform of the spawned object, after selecting the object. The implementation of this subsection is discussed further in Section 4.5.2.

3.4.3 Shadow variation at different times of day

Shadows are often taken into consideration for the visualization of the neighbourhood being designed. For this study since we want to be able to have a capacity of the user to see the design in VR, adding a shadow visualization enables us to see the visual effect of shadows on the neighbourhood that is designed. The user needs to be able to change the time of the day to visualize changing shadows from the objects they place into the AoI. Unreal Engine handles the position of the sun by exposing the time of the day to be edited using a User Interactive widgets. As the position of the sun changes, the gaming engine interprets the light rays to be incident from the updated location and simultaneously updates the visualization of the shadows as well as the colour of the sky due to changing position of the sun. Since the sun's position in the sky changes depending on the spatial coordinate it is being viewed from, the Cesium SunSky actor is utilized to visualize the sun position from the location it is being viewed from.

3.5 EXPORT OF USER-DESIGNED MODEL

After the user design has been made, it is required for the platform to be able to store the locations and orientation so that it can be compared to other designs made by other users and can also be easily exported to other users. The comparison of different designs made by various users is not handled by the platform but could be done by loading the models on simultaneously running instances of the platform on different devices. In addition to the ability to export the model to other users, the saved model should also be usable for further modelling and updating of the design.

The 3D objects spawned by the user at varying locations and their updated positions and orientations are exposed to a json file which contains the ID of the spawned object as well as the location and orientation. The 3D objects themselves are called at runtime to extract their vertices and faces (formed by triangles) to convert them to a glTF file using the obj2gltf library from CesiumGS, which are stored locally. This json file is then utilized, in combination with the extracted faces and vertices to create an instanced 3D model using the Geopipe gltf2i3dm tool. Instanced 3D tiling format is used for the user designed neighbourhood since it enables us to store less information about individual spawned object. The information about the geometry of the 3D objects is stored only once and the instancing of the objects enables use to have the users design to be stored and shared elsewhere. The instanced tiling format can only store one geometry as a glTF file or blob and thus multiple instanced models are required for each type of 3D object placed in the AoI by the designer. These instanced models are then composite together in the composite tiling format which enables ease of exchange of the design by incorporating all tilesets within itself.

This procedure also enables us to incorporate multiple users designing over their own AoI simultaneously, since each one of them can export their designs from their respective work environments and then share the design. Additionally, the users design can also be exported as a PostgreSQL database containing the location, rotation and scales of each type of 3D object utilized by the user. Thereby we are able to answer our research sub-question regarding the dissemination of user-made design along with the contextual 3D city model.

From the composite tiling format, it is possible to extract individual tiles incorporated into the .cmpt file to be able to extract geometry information from the composited tilesets. Furthermore, from the instanced tiles, the geometry information can be extracted as well as the world transforms of each feature type stored in the instanced model. These extracted geometries and their transforms are then utilized to view the model and continue editing them.

3.6 VR BASED VISUALIZATION

For an enhanced user experience, VR proves to be a tool that can enable the users to convey a deeper sense of meaning to their design [Oudshoorn, 2018; Leeuwen et al., 2018; Zhang and Moore, 2014; Axford et al., 2007] as they can do a full virtual walkthrough of the user-made design. The gaming engine employed in this study handles the creation of our application in VR as well as the user interaction with the head mounted display and the VR controllers. This incorporates the processing required to be done to convert a 2D display of the application to a stereographic view where the single camera in case of a non-VR visual is converted to a two-camera setup, with the cameras roughly the same distance apart as the distance between the eyes of a human. The gaming engine handles the post-processing required to be done with the output images to correctly visualize the application without straining the eyes of the user.

3.7 CONCLUSION

In this chapter we discussed the methodological details entailed in this study. The requirements expected of the the final neighbourhood design platform were discussed. Owing to the dependency of the platform developed in this study on the Cesium for Unreal Engine, some possible issues could arise, some of them being the inability of the Cesium plugin to natively support the export and import of instanced and composite tiling formats. Thus for the platform to function, these features will have to be implemented in the backend of Unreal Engine. In Chapter 4 the implementation details are discussed, elaborating upon how the platform is made complete and integrated, fulfilling the requirements set for the platform to be developed as discussed in this chapter.

4 IMPLEMENTATION

In this chapter, we will discuss the implementation utilised for this study, following the methodology discussed in Chapter 3, where the requirements and the framework were established. Beginning with Section 4.1, discussing on the software programs and datasets used and followed by subsequent sections describing the implementational details of each of the steps detailed out in methodology.

4.1 DATASETS, SOFTWARE AND HARDWARE USED

Table 4.1 summarizes the datasets and softwares required and used for this study. The 3D city model is downloaded from 3D BAG, both as a PostgreSQL dump file, for the entirety of the Netherlands and also as CityJSON files for some tiles in the area of the City of Delft to create a small subset for testing purposes. Furthermore, as discussed in Section 3.3, we need to load the proposed land use plan of the AoI, thus the proposed land usage is also required. Moreover, we need to build a library of 3D assets which will act as the user's 3D objects for designing the neighbourhood, however also providing the user with the option to feed in their own 3D elements.

The hardware used to develop the platform consists of a computer with Intel i7-10750H CPU clocked at a base speed of 2.6GHz, with 16 GB of 2933 MHz random access memmory and a 4 GB NVIDIA Quadro T-1000 Max Q graphics processor. Git and Github were utilized to maintain version control over the developed platform and for the report. The platform can be found at this repository: https://github.com/pratyush1611/3dTilesThesis.

4.2 PREPARING CONTEXTUAL 3D CITY MODEL

4.2.1 From PostgreSQL

After downloading the PostgreSQL data dump of 35 Giga Bytes, and using pgrestore command on PostgreSQL to restore the database, it is noted that the database has 3 types of LoD for the buildings made until 2014 in the land registry of the Netherlands. Each LoD has its own table when the data dump is restored. For the purpose of this study only a single LoD is used.

Upon implementation of the pg2b3dm tool from Geodan, it is noted that the tool does not work on our dataset, since it expects the geometry of the data to be in the format of POLYHEDRAL Z, whereas, in the case of the dataset obtained from 3D BAG, we get a geometry type of MULTIPOLYGON Z. To circumvent this issue, a modified version of the pg2b3dm tool is used which works on MULTIPOLYGON Z geometry type. This modification was done by a the 3D geo-information group of TU Delft to utilize the Geodan pg2b3dm tool with the 3D BAG dataset.

4.2.2 From FME 3D tiles writer

Owing to limited memory capacity of the device on which the platform was made, as an alternative to the entire 3D dataset of the Netherlands, a smaller dataset for

Dataset and Software	Reason
3D City model of the Netherlands	To load it in as the context model in 3D tiles format, obtained from 3D BAG
Proposed land use plan	To display in the background while designing neighbourhood, obtained as WMS layer NL.IMRO.0503.SV0001-2001 from Ruimtelijke Plannen
3D objects	To be used for creating the user-made neighbourhood
Unreal Engine	As the platform where the application is developed
Geodan pg2b3dm tool	For conversion of PostgreSQL database to b3dm batched model
GeoPipe gltf2glb and gltf2i3dm tool	For conversion of glb object with json instancing to i3dm tileset
QGIS	To visualize intermediary results from data sources
Safe FME	For conversion of 3D objects to batched model in required reference system
VS Code	For reading through the tileset.json file formed and other files
Steam VR	To interact with the VR controllers inside the application
Cesium ion	To create a web-hosted batched model
Git(Hub)	To maintain a version controlled environment for developing the platform as well as the report
HTC Vive VR Headset and Controllers	To utilize the VR capability of the developed platform and for testing in VR

Table 4.1: Datasets and Softwares used

the city of delft is created for continuing with the remaining steps. This is achieved by first, reprojecting the CityJSON models of Delft, which are in EPSG:7415 to EPSG:4978, and then converting this to a batched model using FME. The process undertaken is highlighted in the Figure 4.1

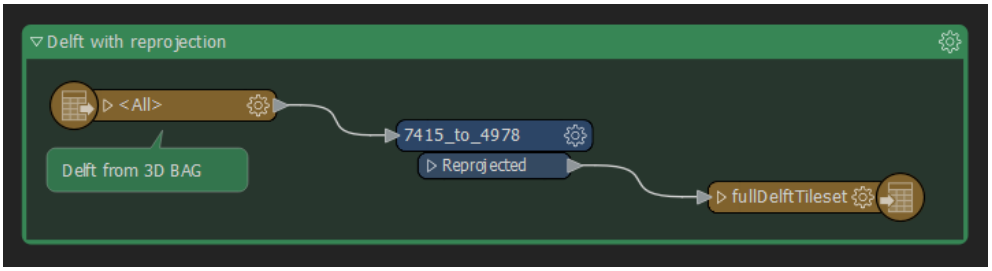


Figure 4.1: FME for creation of 3D tiles

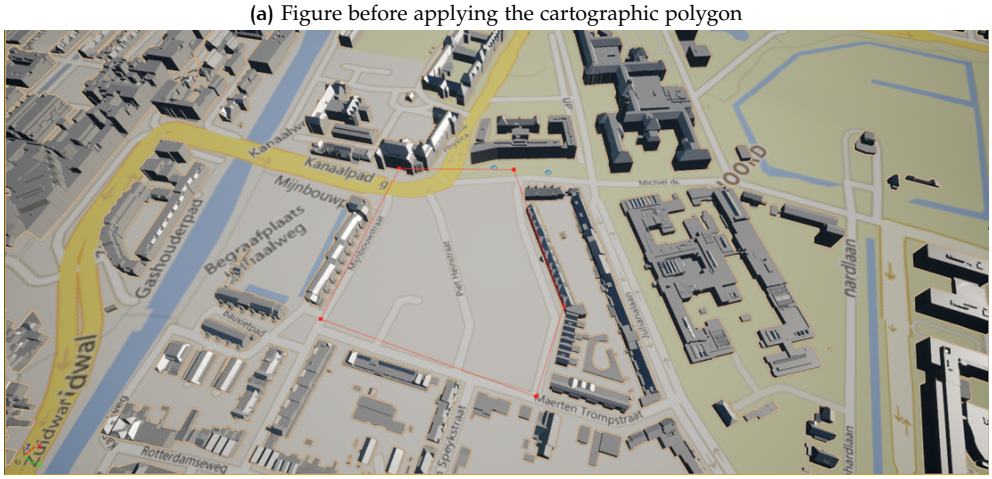
This smaller dataset for city of Delft is hosted locally using the file:/// Universal Resource Locator (URL) or using Cesium ion's web hosting feature. For the case of this study we will use the locally hosted dataset, thus eliminating any chance of the loading failure due to internet failure.

4.3 CLEANING VISUALIZATION FOR OBJECTS WITHIN THE AREA OF INTEREST

In case of CesiumJS, the JavaScript based loader of 3D tiles from caesium, we are able to alter the visualization of the 3D tileset using a show parameter, using which the visualization can be altered on the basis of the batched table containing the attributes of each of the objects within the tiles, however, in case of the setup being used for this study, the plugin Cesium for Unreal Engine does not yet support the

visualization of the tiles according to a show parameter. The plugin, however, does support the ability to draw a polygon and then remove everything within the drawn polygon using Cesium Cartographic Polygon which can be added to the tileset as a component, as discussed in Section 3.2. This renders us with the ability to remove objects inside the AoI as in Figure 4.2

(a) Figure before applying the cartographic polygon



(b) Figure after applying cartographic polygon

Figure 4.2: Working cartographic polygon

4.4 LAND USE LAYER AS BACKGROUND

From the WMS documentation of Ruimtelijke plannnen, for the test case of the City of Delft, the dataset needed is of layer NL.IMRO.0503.SV0001-2001, which has the proposed vision plan for 2040. This WMS layer is loaded inside Unreal Engine using the Cesium plugin using the WMS tile layer component from the plugin. Upon user keypress indicating the display of the background layer, the function calling the display of this layer is called. This function creates a new Cesium WMS tile layer component and attaches it to the tiled layer displaying the terrain with the map on top of it. Moreover, the function also sets the material key layer of this component to Overlay1, thereby ensuring that the WMS layer does not interfere with the map being displayed for contextual information on the terrain. Figure 4.3 shows an intermediary result for displaying land use using a WMS. Appendix B displays the steps used inside Unreal Engine to implement the display of land use layer on key-press.

Figure 4.3: Current Land Use (Delft) as displayed using WMS on QGIS

4.5 NEIGHBOURHOOD DESIGN

In terms of neighbourhood design, we consider many different elements, which can be categorised into buildings and spaces. Spaces can be further divided into streets, public spaces such as parks and squares, etc. For the purpose of this study, the developed neighbourhood design platform enables the designer to spawn their own 3D neighbourhood elements into the AoI, thus the components of neighbourhood design are dependent upon the elements fed in by the user.

In the following sub-sections we will discuss the implementation details of loading a user-designed 3D component, spawning it in the AoI, updating their orientation, scale and position and lastly the updation of the sun position to visualize shadows at different times of the day.

4.5.1 Procedural mesh for Runtime loading

Often-times in urban design, we see the usage of a library of ready-made houses being used for making new layouts, which is convenient, however, for this study, it was thought that it would be an addition to let the user be able to input their own designs in addition to the ones made available in the library.

In the gaming engine being utilized for this study, Unreal Engine, real-time loading of objects is not natively supported. However, to facilitate the building of game levels by game designers, it supports the usage of Procedural Meshes: meshes which can be used to procedurally build a level in a game, for example to change the shape of an object if it is hit by the game character, or to put objects at predefined intervals, but only at runtime.

Exploiting this feature of procedural meshes in the gaming engine, and using a C++ library, the Open Asset Import Library [assimp], which can load 3D meshes of the formats involving 3DS, BLEND (Blender),DAE/Collada, FBX, IFC-STEP, ASE, DXF, HMP, MD2, MD3, MD5, MDC, MDL, NFF, PLY, STL, X, OBJ, OpenGEX, SMD, LWO, LXO, LWS, TER, AC3D, MS3D, COB, Q3BSP, XGL, CSM, BVH, B3D, NDO, Ogre Binary, Ogre XML, Q3D, ASSBIN (Assimp custom format), glTF (partial), 3MF as vertices, faces and normals, we are able to load any of these models at runtime in the game, and then it can be attached to a UI to help users import their own models and place it in the design.

The runtime loading can be further explained using Algorithm 4.1. Additionally, Appendix C details out the workings of the run-time loading of models as implemented using blueprints in Unreal Engine, and Appendix A describes how the UI is implemented to enable dragging and dropping of user-designed 3D objects.

Algorithm 4.1: Workings of runtime loading of 3D object

Input: 3D object, spawn location
Output: 3D object loaded at runtime in Unreal Engine at spawn location

```
procMesh ←instantiate empty procedural mesh ;
if procMesh NOT empty then
    foreach component c in procMesh do
        procMesh ← set as Null ;
        destroy component c;
/* load textures                                  */ ;
texture ← get texture data from png file ;
mat ← create dynamic material instance with texture;
set texture parameter value ;
/* load mesh                                      */ ;
array meshArr ← load mesh from file using ASSIMP library;
foreach array element e in meshArr do
    Add procedural mesh component ;
    attach face to procedural mesh ;
    foreach array sub-element e_sub in e do
        vertices ← extract vertices from e_sub;
        triangles ← extract triangles from e_sub;
        normals ← extract normals from e_sub;
        UVs ← extract UV from e_sub;
        Create mesh section from vertices, triangles, normals and UVs;
        Set material mat
```

4.5.2 Update location of user spawned objects

Once the user spawns the 3D object within their AoI, the spawned object is at an unwanted orientation or is not at the precise position as desired. To enable the user to set the correct position and orientation of the spawned object, a gizmo is designed. This gizmo enables the user to access and update the location parameters of the spawned objects by attaching itself to the spawned object indicated by the user. The gizmo as depicted in Figure 4.4 has the three axis to change its position in the 3D coordinates and also has a ring on the Z-axis to enable the user to update the orientation of the selected object. Algorithm 4.2 describes the functioning of the update of world transforms of the selected object using the gizmo. The implementation of this gizmo component in Unreal Engine using blueprints is described in Appendix D.

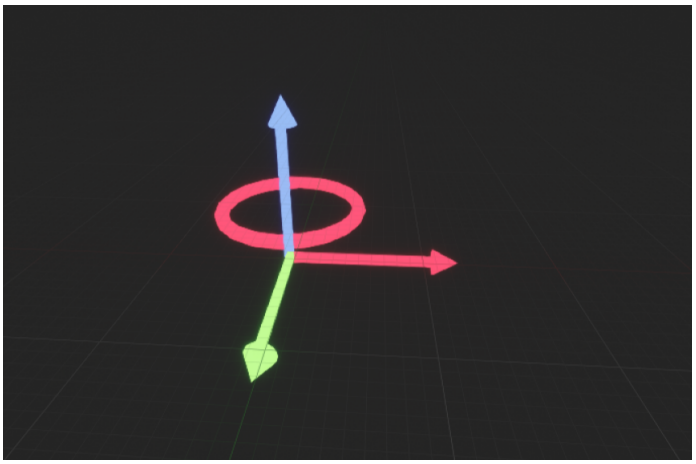


Figure 4.4: Axes on the designed Gizmo

Algorithm 4.2: Pseudocode for updating transform of spawned object using gizmo

Input: keypress event k
Output: gizmo g attached to selected actor

```
1 if k then
2     attach gizmo to spawned object spawnObj ;
3     obj ← detect gizmo axis under user cursor;
4     set gizmo visibility to True;
5     get hit component obj;
6     loc ← get world location of obj;
7     spawnObj ← update location to loc ;
```

4.5.3 Updating sun location

To visualize varying shadows at different times of the day, the position of the sun needs to be updated so that the source of light inferred by the gaming engine changes as well, and thus the gaming engine is able to use its PhysX engine to run ray casting using the updated source of light. To enable this, Unreal Engine provides with different types of source of lights, e.g. point source of light, extended source of light, and spot source. These can be either static or dynamic i.e. movable. For the case of displaying sunlight Unreal Engine also has a Sky source of light known as the sun-sky actor. This sun-sky actor can be modified to take into consideration that the Cesium globe has different coordinate system with respect to the game world. Cesium provides us with the modified sun-sky actor known as the Cesium sun-sky actor, this sun-sky has an exposed functionality to set the value of time of the day. Using a slider widget which is activated by keypress we are able to update the time of the day and thus change the visualization of how the sun and the atmosphere looks as well as the shadows formed by the objects at runtime. The position of the sun is also dependent upon the geolocation from where it is being observed, this is also taken care of by the Cesium sun-sky actor since it places the sun depending on the coordinates from where it is being observed. This implementation in blueprint of Unreal Engine is given in Appendix E.

4.6 EXPORTING USER-DESIGNED MODEL

The 3D object which are loaded at runtime, as described in Section 4.5.1, also have an attached structure which keeps track of their ID, object type, position and orientation. This structure is used to create a json file, which, is then used to create the instanced model using glb2i3dm python script of the Geopipe gltf2glb tool. This tool takes the json schema and .glb (glTF binary) files as input and exports an instanced model of the specified 3D object at specified transformations. The glb file for the geometry is obtained by converting the user input geometry to glb. These processes are elaborated in Algorithm 4.3.

Algorithm 4.3: Pseudocode for export of user-made design

Input: user spawned objects o, button click event k
Output: Files: *design.cmpt* composite tiles, *m.i3dm* instanced models, *pgDB* PostgreSQL database

```
if k then
    foreach object obj of class type userSpawn do
        objGeom ← get geometry information of obj ;
        T ← get world transform of obj ;
        location ← get location information T.location ;
        rotation ← get rotation information T.rotation ;
        scale ← get scale information T.scale
        pgDB ← append to PostgreSQL database storing location, rotation,
          scale ;
        i3dmJSON ← append to JSON file with location, rotation, scale ;
    foreach model type m in class userSpawn do
        m.i3dm ← using gltf2i3dm tool make instanced tiling of m;
    design.cmpt ← using gltf2glb tool composite the i3dm files together for
      ease of sharing;
```

4.7 CONCLUSIONS

In this chapter we discussed the implementation for the various requirements that were discussed in Chapter 3. Firstly the software, hardware and datasets utilized for this study were described, following which the processes of preparation of the contextual 3D tiles, cleaning of visualisation within the AoI, user-enabled neighbourhood design, and export of the design as instanced tiling format have been described. Proceeding this, in Chapter 5, the results obtained from this study are described and discussed upon.

5 RESULTS & DISCUSSION

In this chapter, we discuss the results obtained from our study. This includes the steps taken to create a 3D contextual city model, the removal of objects within the user defined AoI, loading of a planning land use layer in the background, loading of user-made 3D objects and their placement inside the AoI, exporting these designs as instanced model and finally a VR based visualization. Following the results in Section 5.1, in the Section 5.2, the implementations of the obtained results and their inferences are discussed, thereby answering the research questions of this study.

5.1 RESULTS

5.1.1 Loading contextual 3D city model

After following the procedure described in Section 3.1 and Section 4.2, the 3D tiles for the subset of the Netherlands is loaded into the application. This was done both by uploading the subset data as glTF to Cesium ion as well as by converting the CityJSON files to 3D tiles batched model using Safe FME. In Figure 5.1 we see that the CityJSON files are tiled and is viewable in the application, to be used as the context while the user designs their neighbourhood. Moreover, the Open Street Maps 3D city dataset is also made available as visible in Figure 5.1 (b), this dataset is at a lower LoD.

5.1.2 Removing objects within AoI

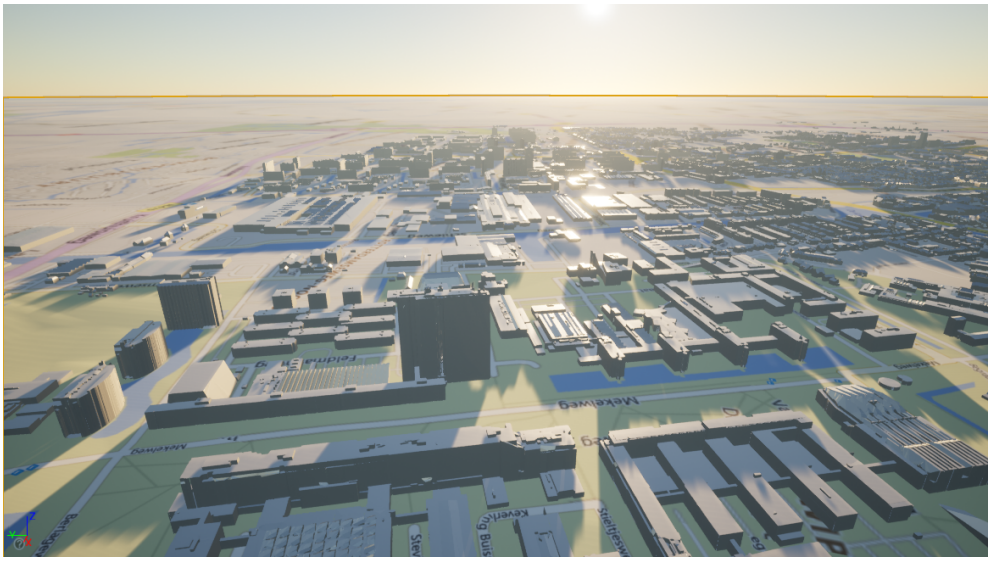
The user defined AoI can be ingested as a geojson or can be made as a polygon at runtime in the application. Once the AoI is ingested, it is attached as a Cartographic Polygon to the 3D tileset from which the visualizations need to be removed. This is seen in Figure 5.2.

5.1.3 Displaying planning Land Use in background

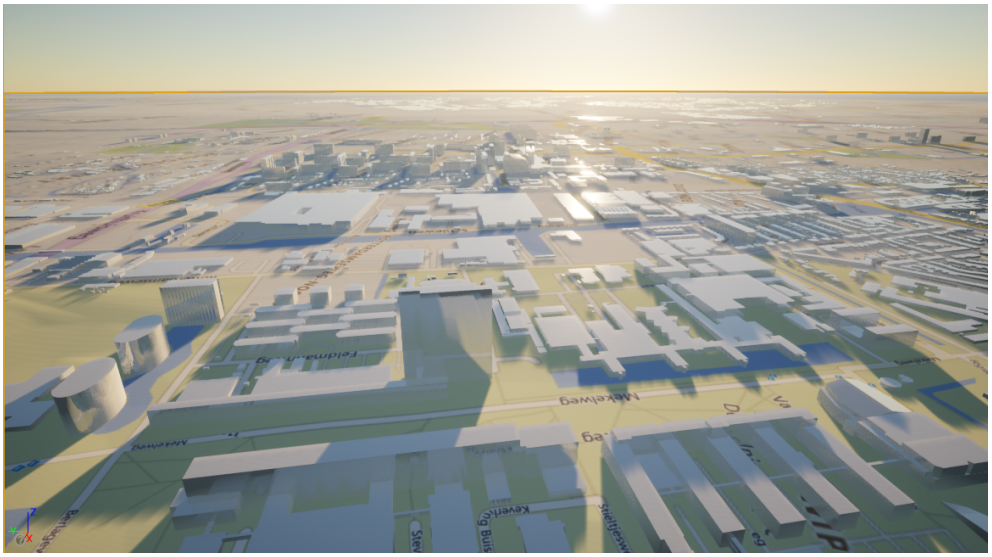
For the user to design the neighbourhood, they are required to make the design which follows predefined zoning regulations or Land Use. This land use is displayed as a part of the terrain on which the contextual city is loaded. In Figure 5.3, the WMS layer for the buildings in 3D BAG is displayed. The same is done for the Land Use plan.

5.1.4 Runtime loading of objects

In our implementation of the application, to load a 3D model which would then be incorporated in the users design, as discussed in the implementation in Section 4.5.1, the 3D object can be loaded by the user itself, and it is processed at runtime to be displayed on the screen. To enable this, a user interface is designed which gives the users options to drag and drop various 3D objects. This UI is visualized in Figure 5.4, the user can drag and drop the 3D objects using the panel on the bottom right, e.g. in Figure 5.4, the buildings are placed and then rotated.



(a) Delft served as 3D tiles using Cesium ion at LoD2.2



(b) Delft and surroundings served as 3D tiles from OSM 3D buildings

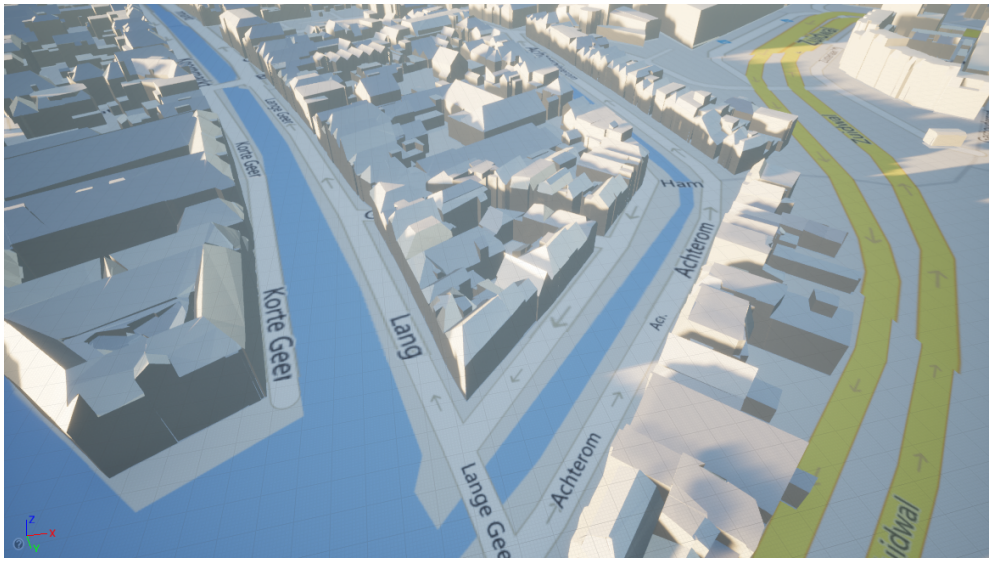
Figure 5.1: 3D Delft as loaded using Cesium ion, and OSM buildings

5.1.5 Updating world transformation of selected object

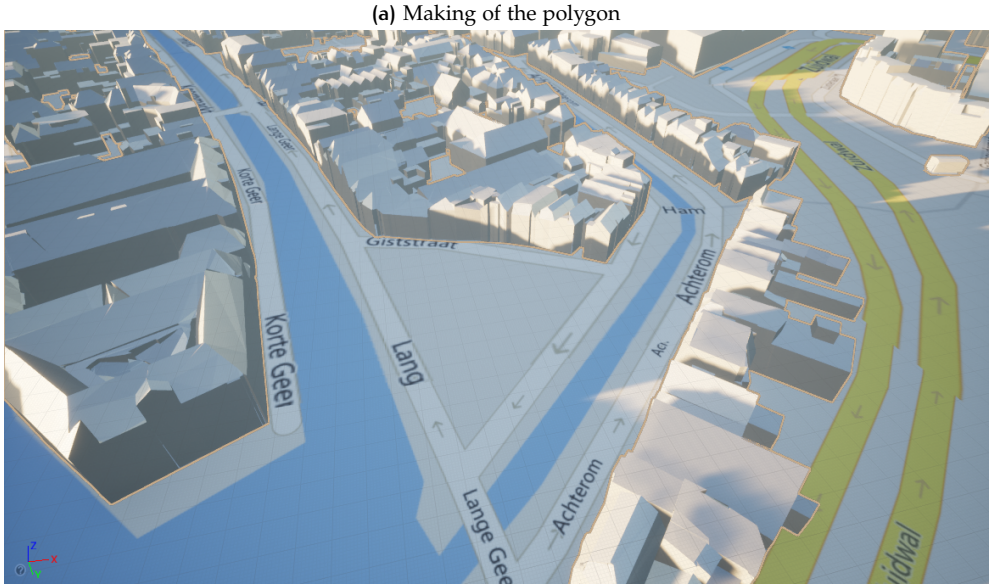
The gizmo implemented in Section 4.5.2 is used for changing the transformations of the 3D object it is attached to, this gizmo, as depicted in Figure 4.4, has three axes and a rotation component, each of which can be clicked by the user to update the position or rotation. This can be seen in action in Figure 5.5, where two objects were spawned into the AoI, and then subsequently their positions and rotation were updated to make them far apart.

5.1.6 Changing shadow visual

Using the implementation discussed in Section 4.5.3, we are able to call the widget and use its slider to update the position of the sun at runtime. This is visualized in Figure 5.6, which portrays three different positions of the sun. The shadows of the buildings are also changes as per the changing source of light, in this case we see that the time is shifted gradually and thus the position of the sun updates and the shadows formed by the buildings also follow the updated location of the sun.



(a) Making of the polygon



(b) Objects within the AoI removed

Figure 5.2: Removal of objects within AoI

5.1.7 Virtual Reality (VR)

Since this research intends to use VR as a visualization tool to enable enhanced user interaction, as discussed in Section 1.2.1, the VR visualization is implemented, as described in Section 3.6. When the user has the VR controllers and headset properly set up and chooses to run the application in VR mode, the game launches a different pawn which has been optimized for handling the VR input and output. Figure 5.7 shows us a scene inside the application where the VR controllers are visible and can be used to run the various options available inside the application.

5.2 DISCUSSION

5.2.1 Visualizing city context in 3D and Land use layer

In the loading of the contextual 3D tiles, the visualization is dependent upon the type of refinement procedure being applied to the tileset. This can be either replace-

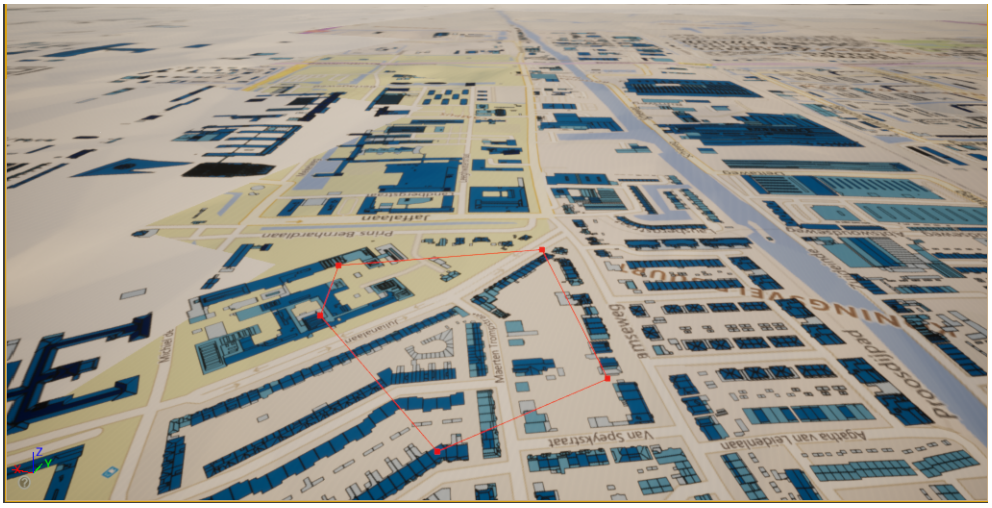


Figure 5.3: WMS layer projected on the terrain in application

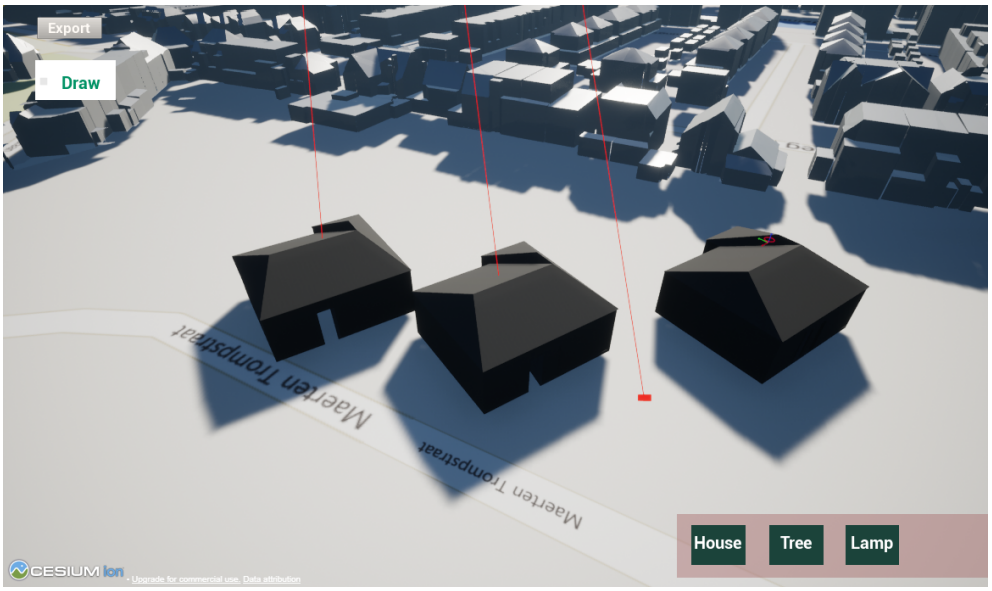
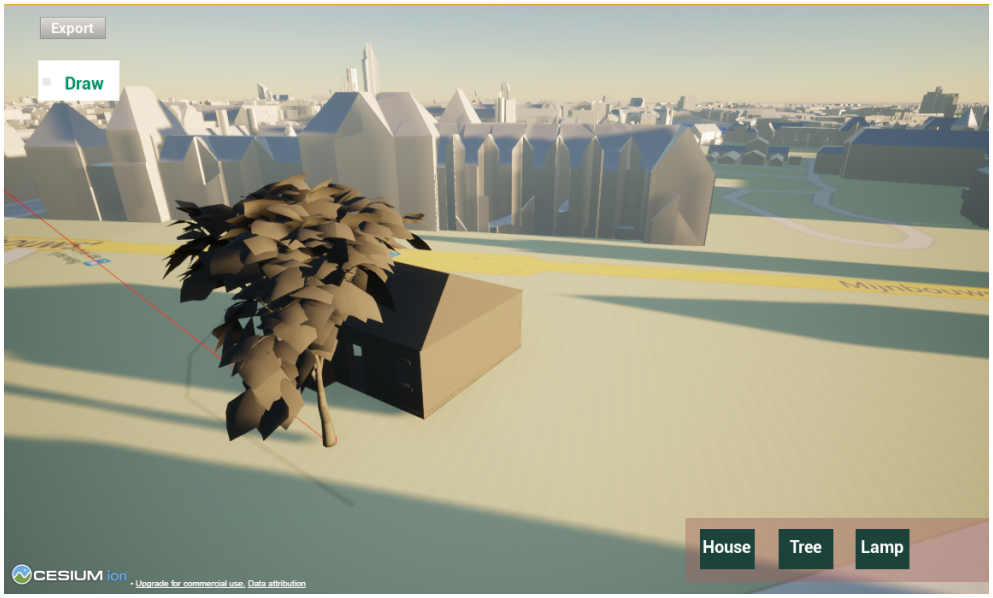


Figure 5.4: UI for drag and drop of user's 3D objects

ment or additive. The difference between these are made clearer in Figure 5.8 and Figure 5.9. In the case of the contextual tiles used for the creation of the application in this study, the refinement strategy used is additive, as can be inferred from Figure 5.10, we see that upon zooming in more tiles are made visible in the viewport. A replacement strategy could be used if the 3D tileset being used has higher LoD buildings at higher zoom and lower LoD buildings in tiles that are viewed from far away.

In our implementation of the removal of objects within the user defined AoI, the Cesium Cartographic Polygon is used and attached as a component to the 3D tileset being served for the city-context. Another way of achieving this could be to use the extensions available in the OGC 3D tiles specifications, which allows for the visualization to be altered by providing with the ID of the objects to be removed. This could have been possible by doing a topological intersection of the AoI with the bounding volume of the 3D tiles starting from the root tile, thus we would have known the exact tiles within which the AoI falls. After the intersection the objects within the selected tiles would be topologically intersected with the delineated AoI and the extension in the tileset.json file would have been added to stop the visual-



(a) Spawned objects

(b) Updated transforms of spawned objects

Figure 5.5: Updating the transforms of spawned objects using gizmo

Figure 5.6: Shadows at differing times of the day

ization. However since the plugin: Cesium for Unreal Engine does not support this yet, the implementation to use the Cesium Cartographic Polygon has been used.

The loading of the WMS layer for land use display could also have been implemented by using 3D tiles, though this would mean that the user needs to convert the planning land use layers to 3D tiles. In the implementation chosen for this study, using WMS, the user is free to use any service which serves them with WMS, thus eliminating the need for the user to know the process of tiling of the land use layers, however this can be implemented as a separate option where the user ingests the Land Use layer. Additionally, the user could also be given the option to use their own images tiled using a Geo-web server and then served as a WMS or a Web Map Tiling Service (WMTS), thereby giving the user complete control over what

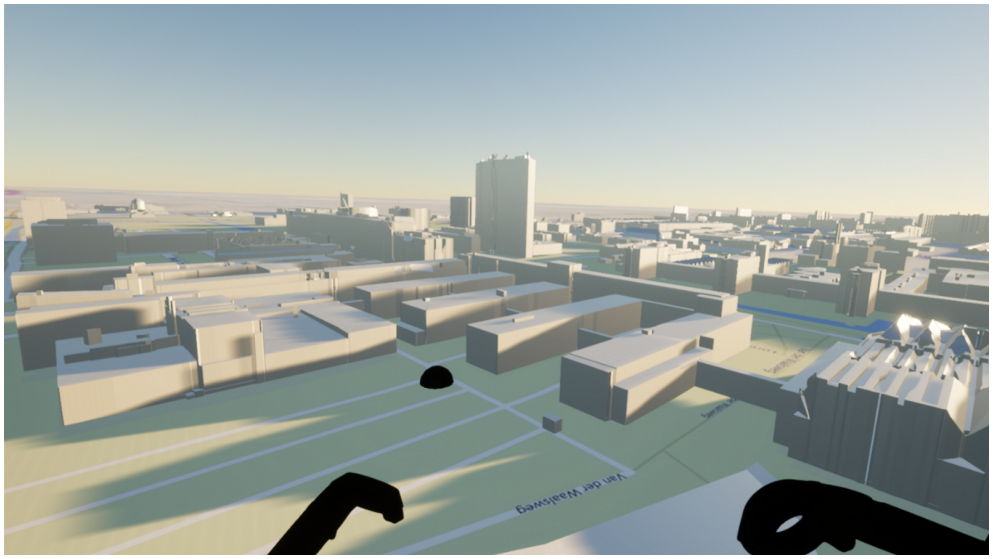
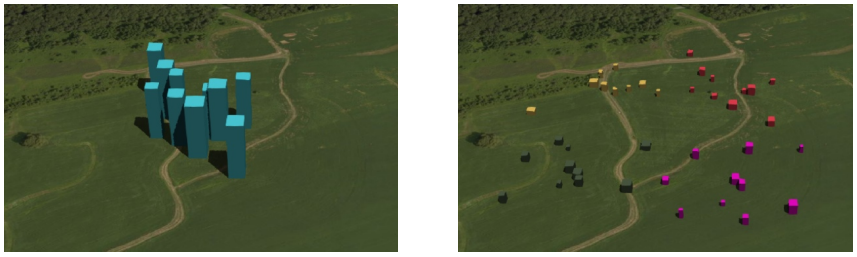
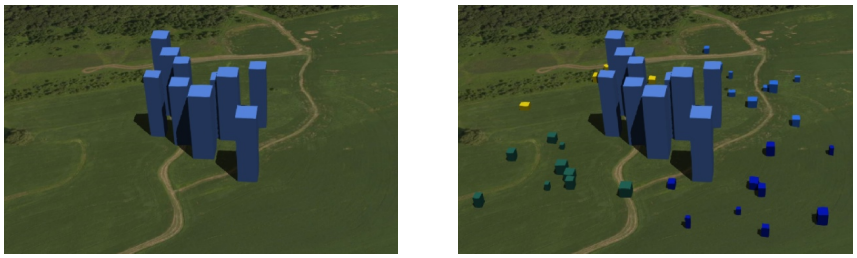


Figure 5.7: Application run in VR using HTC Vive controllers



(a) Parent **(b)** Refined

Figure 5.8: Replacement refinement strategy (source: [Open Geospatial Consortium, 2019])



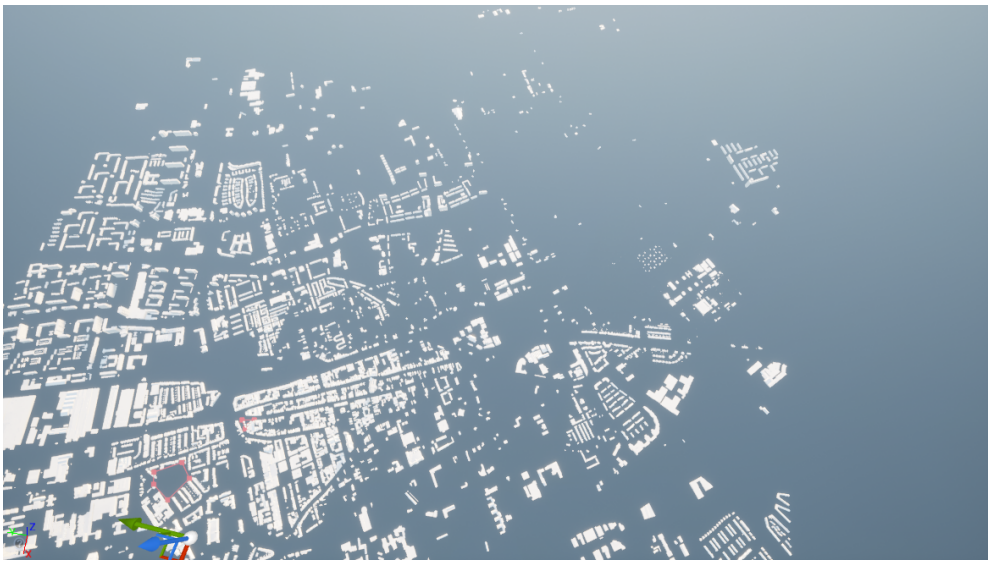
(a) Parent **(b)** Refined

Figure 5.9: Additive refinement strategy (source: [Open Geospatial Consortium, 2019])

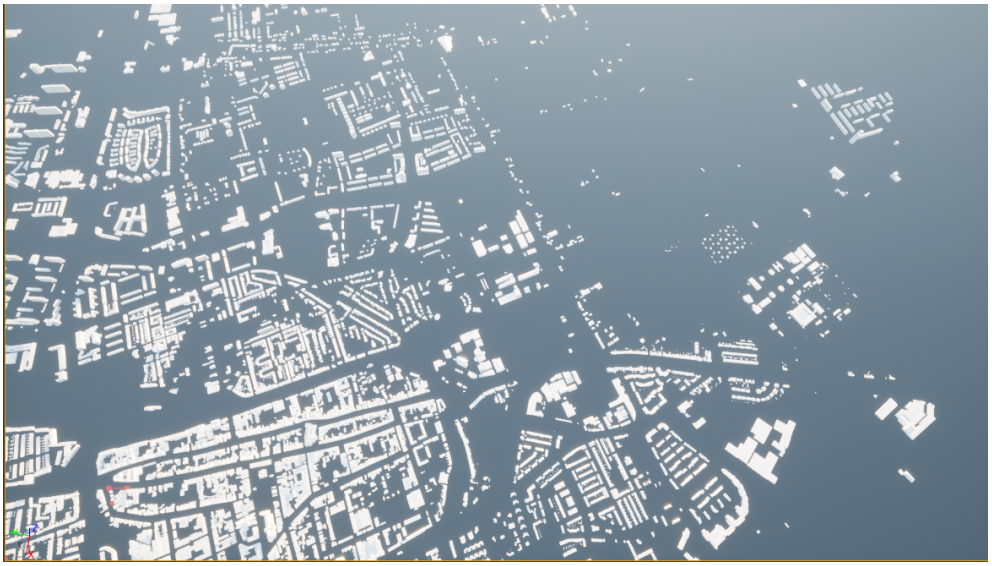
background layer they want to visualize instead of a Land Use plan served by the planning authority.

5.2.2 User-made neighbourhood design

The implementation for loading 3D objects in this thesis enables user to feed their own designed objects at 1:1 scale. This facilitates the user to design their objects in any desired LoD. In this research, for testing purposes, a LoD4 model of the house was implemented and imported into the design AoI. The usage of the gaming engine allows for the user to enter the LoD4 building and traverse through it with proper collisions against the wall surfaces. The platform could also be further developed to allow the designers to place interior design elements within these spawned objects,



(a) Tiles as seen zoomed-out



(b) Zoomed-in

Figure 5.10: Refinement as seen in developed application

thereby becoming an integrated platform for not only urban planners and designers but architects and interior designers as well.

5.2.3 Models as 3D tiles

The user-made design is exported as an instanced 3D tiling format (*.i3dm*). This enables the user to share the design to other designers who might want to change or add other features to the model. Currently, the Cesium for Unreal Engine plugin does not support loading of i3dm files, however the development is in pipeline, and soon it should be able to load instanced models. However, in the backend of the application, we can easily extract the geometry information from the i3dm file as well as the positions, rotations and scaling of each of these instances of the geometry using the i3dm.tooling tool from Geodan, thereby enabling the application to load the 3D model from a saved i3dm file or from another user.

In neighbourhood design, the ability to have the contextual city model is beneficial to the designer who is designing the neighbourhood area plan. The context gives the ability to design the neighbourhood with the textural facades that match the context. Moreover, after the designing phase of the plan, if some analysis needs to be done on the user-designed model, then the context can also be visualized in a form of heatmap, along with the users' design e.g. solar potential analysis, energy demand analysis, etc.

5.2.4 LoD while designing

The context of the AoI can be viewed in multiple LoD. While designing the neighbourhood, if the users own 3D objects are in a lower level of detail then it is better to have the contextual buildings in a similar or lower level of detail. In the case of 3D tiles the visualization of the tiles is maintained as Hierarchical level of detail, meaning that the LoD reduces as the tiles are away from the view frustum, thus enabling a seamless transition of the view of the contextual city buildings.

5.3 CONCLUSION

In this chapter, the results of this study was discussed, wherein the obtained functionality were shown, starting from loading of contextual city model, loading of a planning base layer in the background, ingestion of an AoI, loading of user-designed 3D assets and spawning these assets, followed by the mechanism to update the transforms of these spawned objects and lastly the updating sun position on basis of time of the day and VR based visualization. These results are illustrated by the means of figures in this chapter. In Section 5.2, the underlying working principles for the case of loading of land use layer, user made design, usage of user-made design as 3D tiles and the working of hierarchical level of detail are discussed. In the following chapter, this study is concluded and its research questions are answered along with a discussion on the limitations and future scope.

6 CONCLUSIONS, LIMITATIONS & FUTURE SCOPE

This chapter concludes this research and proposes the future scope of this study. In the Section 6.1 the research overview is discussed wherein the research questions established in Chapter 1 are answered based on the research done in this study. Following conclusions, the future scope of the study is deliberated in Section 6.3, discussing further implementation and use cases of the platform developed in this study.

6.1 CONCLUSIONS

This thesis was aimed at developing a way to incorporate the OGC 3D tiling technique into a platform meant for enabling urban designers to dynamically design urban neighbourhoods in a given AoI in a VR environment. This was done in order to facilitate a platform where the neighbourhood designs could be made not only in 2D but in 3D as well, with much more enhanced user experience as it is VR enabled, thus also giving the capacity for a better stakeholder engagement in terms of urban planning. To achieve this aim, a main research questions and subsequent sub-questions were designed; these questions are reviewed below, first answering the sub-questions, followed, in the end, by the main research question.

a. Can the 3D tiling technique be utilized for the use case of urban neighbourhood development?

The results discussed in Chapter 5 shows that the developed application utilizes the 3D tiling technique in the use case of urban neighbourhood designing. The contextual 3D city model is loaded as a batched 3D model served either locally or through a web-server. This can be at different desired levels of detail, as seen in Figure 5.1. Moreover, the tiling technique ensures that the 3D city model is loaded with minimal load on the client-side, since only tiles which are within the view frustum are loaded from the web-server and tiles which are further away are displayed with a lower LoD. The hierarchical LoD ensures that the tiles which are further away from the camera, are displayed only until certain extent, defined by the geometric error of the tileset, thereby not loading all the leaf node tiles in the tileset being viewed at a distance further away from the camera. This applied not only to the contextual city model but also the user-designed neighbourhood.

b. In the use case of urban neighbourhood design, how can dynamic user selected designs be effectively visualized?

For the purpose of visualizing the contextual 3D city model along with the dynamically user-made design of the neighbourhood, results of this study show that it is possible for viewing the batched model of the 3D contextual model along with the user-made design which can be in form of an instanced model or a PostgreSQL database containing the positions, rotations and scales of each instance of the different models employed by the user in their designed neighbourhood.

For visualizing the user-made design, along with the ability to further edit the designed model, the geometries of the 3D objects are extracted from the instanced

model or from the PostgreSQL database along with the geometries provided separately, and the geometries can be loaded in a format native to the application in form of procedural meshes which are built by vertices, triangles and normals. Whereas, for the purpose of only visualizing and not further editing the models, the instanced model and the batched model can be loaded simultaneously, though this is not possible at present using the Cesium plugin in Unreal Engine, other 3D tiles viewers can load multiple tile formats simultaneously e.g. CesiumJS, Mapbox GL JS or three.js and support the visualization of instanced 3D tiling format. Moreover, the user-made designs can be incorporated into the tileset representing the batched 3D model using the glTF extensions enabling us to attach one tileset to another; apart from this, the composite 3D tiling format can also be used to store multiple type of 3D tiling formats together in a single file. Though the developed application does not support these abilities of the 3D tiling specification, it is possible to do so using the aforementioned 3D tile viewers.

c. How can user-designed neighbourhoods be dynamically disseminated in the form of 3D tiling to incorporate it with the rest of the contextual 3D tiles?

From this study, it is seen that the user-designed model can be rendered natively as procedural meshes, which is loaded at runtime, allowing for the mesh to be manipulated, such as: updating position in world, rotation or scaling. Whilst the 3D objects are being used for designing the neighbourhood, they are loaded as a mesh inside the application, however, for disseminating this user-made design, the entire design is converted to instanced 3D tiles with each type of object having its own instanced model, since an instanced tiling format only supports a single glTF file/blob per tile. To disseminate these instanced model together, a composite tiling format is used which can hold multiple tiling formats together in a single file with references to other tiling formats.

Additionally, the user-made design can also be disseminated as a batched 3D tiling format model, however, doing so renders us with more data usage since every object is then stored at its position in world and thus creates a redundancy since in the user-made design the objects being used are static, only their rotation position and scaling is changed by the user, the geometry being used is the same for every instance of each 3D object type, thus an instanced model is the most efficient way of disseminating the user-made model as 3D tiling.

Main Question: To what extent can we make a combination of existing 3D city datasets with a user interactive neighbourhood designing platform in a virtual reality environment?

Drawing from the research sub-questions discussed above, it can be concluded that the existing 3D datasets can be combined with user-made designs placed inside an interactive application which allows for the user to place their own 3D models inside the design, which can be in over 30 different 3D modelling formats. These models designed by the user could be in the any possible level of detail, and thus it is also possible for the user to actually do a walkthrough of not only the designed neighbourhood but also each object placed in the design, such as a LoD4 house in which the user can then enter and also place other objects such as furniture or lighting.

Furthermore, the user-made design can be easily exported as a composite 3D tile containing the user-designed features as instanced models, only storing a single instance of each type in a binary glTF format along with their transformations. Accompanying the composite tile, each features' instanced tiles are also made available and these tiled filed can then be shared with other users for viewing or further editing or even for analysis on the designed neighbourhood.

The VR capability of the application empowers the user to have an amplified experience of the designed neighbourhood. The textures of the models designed by the users can also be attached to the instanced model formed and thus a high quality render of the designs can be visualized by the user and the stakeholders involved. VR based designing also gives the user a greater sense of understanding of the neighbourhood they are designing in terms of how it will look like, e.g. the feature for shadow visualization can be of importance to the 3D model designer as well as urban planner since it will show them an estimated idea of how the design is going to feel like for a person who will inhabit the designed neighbourhood.

6.2 LIMITATION

Although this thesis has shown that 3D tiling technique can be effectively utilized for disseminating existing 3D data with an interactive and dynamically design neighbourhood, there are certain limitations to the existing workflow used for this study; outlined as follows:

- **Coordinate reference system:** The 3D tiles being served for the application is loaded into the application using the Cesium for Unreal Engine plugin. This plugin is a work in development and is open-source, meaning that there are constant developments being made by the community supporting and utilizing the plugin for various purposes. Unreal Engine, being a platform for designing games, does not natively support the coordinate reference systems that we often use in geospatial systems. To counter this the Cesium plugin uses a ECEF reference frame with the centre of mass of the WGS84 ellipsoid as its origin. The plugin thereby, supports the loading of geospatial data inside Unreal Engine by transforming the coordinates on the fly, using a Cesium georeference anchor which allows for the visualization of a 100 KM radius around the placed anchor. Beyond this radius, the geoid shape of the earth comes into play and another georeference anchor needs to be placed. However, for our purpose of designing and visualizing neighbourhoods of urban scale this is not a problem since most neighbourhoods are within the size of 10 sq. km.

  The contextual 3D model served by the web-server to the application however needs to be in the EPSG:4978 coordinate system with the heights relative to the WGS84 ellipsoid for them to be displayed at the right heights and clamped to the ground. In case the 3D model is hosted using Cesium ion, then the coordinate reference system is transformed by Cesium ion on the server side, additionally it also provides the user with the option to manually adjust the location of the 3D dataset.

- **Instanced and composite tiling format:** Though the application is able to convert the user-designed model of the neighbourhood into both instanced tiling formats for each type of 3D object and a composite tile containing references to all the instanced tiles, the Cesium plugin for Unreal Engine utilized in this thesis currently does not support the ability to visualize .i3dm and .cmpt tiling formats. Although a native support would enable the users to directly load the model from these formats, the current implementation extracts the geometries from these models and converts them to geometries native to the gaming engine, for the user to further edit the model.

- **Support for extensions in the batched model:** The OGC 3D tiling specification enables for multiple tilesets to be referenced from a single tileset. There are extensions provided in the batched tiling format which can be used to hold multiple contents within a single batched model but with references to other content types such as instanced 3D tiling format. These extensions can also be

used for selectively visualizing the contents of the tileset, thereby eliminating the need to create a cartographic polygon for removal of objects within the users' AoI. These extensions are still under development for the Cesium for Unreal plugin and thus cannot be incorporated in the application developed for this thesis.

6.3 FUTURE SCOPE & RECOMMENDATIONS

As a part of the discussion in Section 5.2, recommendations on what could have been done further or differently in the implementation is deliberated. The recommendations for improvement of the platform developed as part of this thesis as well as other scopes are discussed as follows:

- **Real-time datasets:** The platform designed as a part of this thesis uses 3D tiles to display the contextual 3D city model. It also supports the use of WMS and WMTS layers which can be displayed on the terrain. These tiling services can be used to display real-time data obtained from sensors in the city for a virtual monitoring of different parameters of the city, e.g. air quality, temperature, rainfall, water logging, etc, to provide with a more enhanced and engaging digital twin of the city. Whilst the real-time sensor datasets are not necessarily available as WMS or WMTS, they might be available through other standards, the data can be processed in the backend and served as a WMS or WMTS.
- **Urban simulations:** In addition to real-time visualization of sensor datasets, the platform could also be used to implement urban simulations on the designed model, e.g. traffic simulation of the designed neighbourhood along with the contextual model, solar potential analysis of the designed neighbourhood, visualization of simulated land values depending on the design and the neighbouring land values.
- **Centralised 3D data dissemination architecture:** In the platform developed in this research, the 3D tiled user-made design is exported and stored locally on the device. As an additional feature, the platform could also be linked to a centralised repository which would store the designs made by different users on different devices. This repository would then act as a source to disseminate the user-made designs to other platforms which have the capability to read instanced and batched tiling formats.
- **User Interface:** The User Interface of the application designed could be improved to make the users experience of the tool seamless. There are researches on effective visualization of UI in a VR environment, to have minimal stress on the eyes of the user who is using the VR controllers and headset in the platform.
- **Replacement Strategy:** The replacement strategy used for the contextual 3D city model could be changed such that the tiles viewed far away display only buildings in a lower LoD. For the implementation in this study, the strategy used is additive, but this could further be studied upon to ensure the most optimal loading of 3D tiles.

Apart from these recommendations for the improvement of the platform developed in this research, several other research ideas and questions also come up while framing the research questions, but which have not been included owing to them being vast topics of discussion and research themselves. These research ideas are described as follows:

- **Optimization of tiling technique:** The tiling technique used for this research is based on the Safe FME 3D tiles writer, which uses a maximum number of

features displayed in a tile as the threshold for creating the tiles. Another tiling technique used in this research was based on the adaptive tiling from Cesium ion which utilizes a different tiling technique per tile, depending on the optimization with respect to the amount of time it takes to load the tile. There are other tiling techniques which could be used for 3D tiling as well, e.g. using a quadtree, octree, kd-tree, adaptive kd-tree, etc and this can be an area of research itself, about the best technique of tiling the 3D models.

- **Automated design of urban neighbourhoods:** There have been research on parametrically forming urban neighbourhoods, especially in the case of game level designing, wherein an algorithm automatically builds a neighbourhood for the user to select from and then edit. This could also be implemented using 3D tiles, since 3D tiles in Unreal Engine enables collision of objects in the game with the 3D tiles. The research would then be on how automatically created neighbourhoods can be effectively disseminated using 3D tiles and could also explore on the implementation of editable batched models, apart from the usage of instanced models.

- **Effectiveness of VR in urban design:** This thesis uses VR as a tool for visualizing and designing urban neighbourhoods. While multiple researches show that VR enables enhanced stakeholder participation, a research on the effectiveness of VR as a designing tool could be done, exploring in depth, by means of surveys and interviews with stakeholders and designers, the effectiveness of VR for designing in 3D.

BIBLIOGRAPHY

Geopipe. gltf2glb. https://github.com/Geopipe/gltf2glb, 02 2022. URL https://github.com/Geopipe/gltf2glb.

Firas A. Salman Al-Douri. Impact of utilizing 3d digital urban models on the design content of urban design plans in us cities. *Texas AM University*, 10 2006. URL https://oaktrust.library.tamu.edu/handle/1969.1/4324.

Geemente Amsterdam. 3d amsterdam — v2.9.1. https://3d.amsterdam.nl/web/index.html#over, 2021. URL https://3d.amsterdam.nl/web/index.html#over.

Stephen Axford, Garry Keltie, and Christine Wallis. Virtual reality in urban planning and design. *Multimedia Cartography: Second Edition*, pages 283–294, 2007. doi: 10.1007/978-3-540-36651-5_20. URL https://link.springer.com/chapter/10.1007/978-3-540-36651-5_20.

P Ayres. *Persistent modelling: extending the role of architectural representation*. Routledge, 2012. URL https://books.google.com/books?hl=en&lr=&id=woOoAgAAQBAJ&oi=fnd&pg=PR1&dq=Persistent+Modelling:+Extending+the+Role+of+Architectural+Representation&ots=3EphEDWAoT&sig=lLRxPOw9cJ7T9qtOzNvdH1VNe-8.

3D BAG. Overview - 3d bag. https://docs.3dbag.nl/en/, 3 2021. URL https://docs.3dbag.nl/en/.

Gonzalo Besuievsky and Gustavo Patow. Procedural modeling historical buildings for serious games. *Virtual Archaeology Review*, 4:160–166, 11 2013. ISSN 1989-9947. doi: 10.4995/VAR.2013.4268. URL https://polipapers.upv.es/index.php/var/article/view/4268.

Filip Biljecki, Jantien Stoter, Hugo Ledoux, Sisi Zlatanova, and Arzu Çöltekin. Applications of 3d city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4:2842–2889, 12 2015. ISSN 22209964. doi: 10.3390/IJGI4042842. URL https://www.mdpi.com/2220-9964/4/4/2842/htmhttps://www.mdpi.com/2220-9964/4/4/2842.

R. Billen, A.-F. Cutting-Decelle, O. Marina, J.-P. de Almeida, Caglioni M., G. Falquet, T. Leduc, C. Métral, G. Moreau, J. Perret, G. Rabin, R. San Jose, I. Yatskiv, and S. Zlatanova. 3d city models and urban information: Current issues and perspectives - european cost action tu0801. *EDP Sciences*, pages I–118, 2014. doi: 10.1051/TU0801/201400001. URL https://3u3d.edpsciences.org/articles/3u3d/abs/2014/01/3u3d-cost2013/3u3d-cost2013.html.

G. Buyuksalih, S. Bayburt, A. P. Baskaraca, H. Karim, A. Abdul Rahman, G. Buyuksalih, S. Bayburt, A. P. Baskaraca, H. Karim, and A. Abdul Rahman. Calculating solar energy potential of buildings and visualization within unity 3d game engine. *ISPAr*, 42W5:39–44, 9 2017. ISSN 2194-9034 The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. doi: 10.5194/ISPRS-ARCHIVES-XLII-4-W5-39-2017. URL https://ui.adsabs.harvard.edu/abs/2017ISPAr42W5...39B/abstract.

Cesium GS Inc. 3d tiles. https://prismic-io.s3.amazonaws.com/cesium/5f705923-8ff1-410e-990a-0018157e8086_3d-tiles-overview.pdf, 2020. URL https://prismic-io.s3.amazonaws.com/cesium/5f705923-8ff1-410e-990a-0018157e8086_3d-tiles-overview.pdf.

Ran Chen. The development of 3d city model and its applications in urban planning. *Proceedings - 2011 19th International Conference on Geoinformatics, Geoinformatics 2011*, 2011. doi: 10.1109/GEOINFORMATICS.2011.5981007.

CSIRO. Nicta/cesium-vr: Plugin for cesium web-based virtual globe software to support the oculus vr headset. https://github.com/NICTA/cesium-vr, 5 2016. URL https://github.com/NICTA/cesium-vr.

Paul Cureton and Nick Dunn. Digital twins of cities and evasive futures. *Shaping Smart for Better Cities*, pages 267–282, 1 2021. doi: 10.1016/B978-0-12-818636-7.00017-2.

Peter H. Dana. Geodetic datum overview. https://web.archive.org/web/20180118151658/https://www.colorado.edu/geography/gcraft/notes/datum/datum.html, 1995. URL https://web.archive.org/web/20180118151658/https://www.colorado.edu/geography/gcraft/notes/datum/datum.html.

Ruben de Laat and Léon van Berlo. Integration of bim and gis: The development of the citygml geobim extension. *Springer*, pages 211–225, 2011. doi: 10.1007/978-3-642-12670-3_13. URL https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-3-642-12670-3_13.

Epic Games. Physics — unreal engine documentation. https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Physics/, 8 2021a. URL https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Physics/.

Epic Games. Virtual reality development — unreal engine documentation. https://docs.unrealengine.com/4.26/en-US/SharingAndReleasing/XRDevelopment/VR/, 2021b. URL https://docs.unrealengine.com/4.26/en-US/SharingAndReleasing/XRDevelopment/VR/.

ESRI. Advanced 3d city design software, arcgis cityengine. URL https://www.esri.com/en-us/arcgis/products/arcgis-cityengine/overview.

ESRI. From cityengine to unreal engine: the journey from first design steps to high-quality real-time visualization. https://www.esri.com/arcgis-blog/products/city-engine/design-planning/the-journey-from-cityengine-to-unreal-engine/, 5 2019. URL https://www.esri.com/arcgis-blog/products/city-engine/design-planning/the-journey-from-cityengine-to-unreal-engine/.

Stephen Ferguson. Apollo 13: The first digital twin — simcenter. https://blogs.sw.siemens.com/simcenter/apollo-13-the-first-digital-twin/, 4 2020. URL https://blogs.sw.siemens.com/simcenter/apollo-13-the-first-digital-twin/.

RUNNAN FU, Yuzhen JIN, ZHENYU LIU, Xenia Una Mainelli, THEODOROS PAPAKOSTAS, and Linjun Wang. 3d representations for visual insight. https://repository.tudelft.nl/islandora/object/uuid%3Aa6c4f703-b048-40e3-9661-be00c0fab804, 2021. URL https://repository.tudelft.nl/islandora/object/uuid%3Aa6c4f703-b048-40e3-9661-be00c0fab804.

Geodan. Geodan/pg2b3dm: Tool for converting from postgis to b3dm tiles. https://github.com/Geodan/pg2b3dm, 1 2022. URL https://github.com/Geodan/pg2b3dm.

GlobalData Thematic Research. History of smart cities: Timeline. https://www.verdict.co.uk/smart-cities-timeline/, 2 2020. URL https://www.verdict.co.uk/smart-cities-timeline/.

Aswin Indraprastha and Michihiko Shinozaki. The investigation on using unity3d game engine in urban design study. *Journal of ICT Research and Applications*, 3:1–18, 2009. ISSN 2338-5499. doi: 10.5614/ITBJ.ICT.2009.3.1.1. URL https://journals.itb.ac.id/index.php/jictra/article/view/180.

Sergey Ivanov, Ksenia Nikolskaya, Gleb Radchenko, Leonid Sokolinsky, and Mikhail Zymbler. Digital twin of city: Concept overview. In *2020 Global Smart Industry Conference (GloSIC)*, pages 178–186, 2020. doi: 10.1109/GloSIC50886.2020.9267879.

Jihong Jiang. objto3d-tiles convert obj model file to 3d tiles. https://github.com/PrincessGod/objTo3d-tiles, 2019. URL https://github.com/PrincessGod/objTo3d-tiles.

Khronos Group. gltf overview - the khronos group inc. https://www.khronos.org/gltf/, 6 2017. URL https://www.khronos.org/gltf/.

Thomas H. Kolbe. Representing and exchanging 3d city models with citygml. *Lecture Notes in Geoinformation and Cartography*, pages 15–31, 2009. ISSN 18632351. doi: 10.1007/978-3-540-87395-2_2. URL https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-3-540-87395-2_2.

Steven M Lavalle. *Virtual Reality*. Cambridge University Press, 2020. URL http://lavalle.pl/vr/.

Hugo Ledoux. Datasets — cityjson. https://www.cityjson.org/datasets/, 2019. URL https://www.cityjson.org/datasets/.

Hugo Ledoux, Ken Arroyo Ohori, Kavisha Kumar, Balázs Dukai, Anna Labetski, and Stelios Vitalis. Open geospatial data, software and standards cityjson: a compact and easy-to-use encoding of the citygml data model. *Open Geospatial Data, Software and Standards*, 2019. doi: 10.1186/s40965-019-0064-0. URL https://doi.org/10.1186/s40965-019-0064-0.

Jos P. Van Leeuwen, Klaske Hermans, Antti Jylhä, Arnold Jan Quanjer, and Hanke Nijman. Effectiveness of virtual reality in participatory urban planning. *ACM International Conference Proceeding Series*, pages 128–136, 11 2018. doi: 10.1145/3284389.3284491. URL https://doi.org/10.1145/3284389.3284491.

Jacob Ragot Onyimbi, Mila N Koeva, and J Flacke. Assessing the impact of 3d visualization and e-participation on public participation in planning processes in kisumu city, kenya. https://webapps.itc.utwente.nl/librarywww/papers_2017/msc/upm/onyimbi.pdf, 2017. URL https://webapps.itc.utwente.nl/librarywww/papers_2017/msc/upm/onyimbi.pdf.

Open Geospatial Consortium. 3d tiles specification 1.0. *Github*, 1 2019. URL https://docs.opengeospatial.org/cs/18-053r2/18-053r2.html.

Remco Oudshoorn. Utlizing virtual reality in a dynamic stakeholder process. http://resolver.tudelft.nl/uuid:4b6fd4f2-7a6f-431f-b181-a8c8d3813ff6, 2018. URL http://resolver.tudelft.nl/uuid:4b6fd4f2-7a6f-431f-b181-a8c8d3813ff6.

David Rogers. Twin tracks: The drive to create a smart digital model of the uk. *Construction Research and Innovation*, 10:49–52, 4 2019. ISSN 2045-0249. doi: 10.1080/20450249.2019.1621591. URL https://www-tandfonline-com.tudelft.idm.oclc.org/doi/abs/10.1080/20450249.2019.1621591.

Geemente Rotterdam. Rotterdam in 3d — rotterdam.nl. https://www.rotterdam.nl/werken-leren/3d/, 2021. URL https://www.rotterdam.nl/werken-leren/3d/.

Uwe Rüppel and Kristian Schatz. Designing a bim-based serious game for fire safety evacuation simulations. *Advanced Engineering Informatics*, 25:600–611, 10 2011. ISSN 1474-0346. doi: 10.1016/J.AEI.2011.08.001.

Ehab Shahat, Chang T. Hyun, and Chunho Yeom. City digital twin potentials: A review and research agenda. *Sustainability 2021*, 13:3386, 3 2021. ISSN 20711050. doi: 10.3390/SU13063386. URL https://www.mdpi.com/2071-1050/13/6/3386.

Milica Stojanović, Petar Mitković, and Mihailo Mitković. The scenario method in urban planning. *Facta universitatis - series: Architecture and Civil Engineering*, 12:81–95, 2014. ISSN 0354-4605. doi: 10.2298/FUACE1401081S. URL http://www.doiserbia.nb.rs/Article.aspx?ID=0354-46051401081S.

Tygron. Tygron support wiki. https://support.tygron.com/wiki/Main_Page, 5 2022. URL https://support.tygron.com/wiki/Main_Page.

Geemente Utrecht. Utrecht in 3d — municipality of utrecht - heritage. https://erfgoed.utrecht.nl/utrecht-in-3d/, 2021. URL https://erfgoed.utrecht.nl/utrecht-in-3d/.

Willem Vletter. First steps in the use of a game engine for historical roads and paths research. *Journal of Computer Applications in Archaeology*, 2:3–11, 2 2019. ISSN 25148362. doi: 10.5334/JCAA.18/METRICS/. URL http://journal.caa-international.org/articles/10.5334/jcaa.18/.

Wikipedia. Earth-centered, earth fixed coordinates. https://en.wikipedia.org/wiki/File:Ecef_coordinates.svg, 9 2021. URL https://en.wikipedia.org/wiki/File:Ecef_coordinates.svg.

Sisi Zhang and Antoni B. Moore. The usability of online geographic virtual reality for urban planning. *Lecture Notes in Geoinformation and Cartography*, pages 225–242, 2014. ISSN 18632351. doi: 10.1007/978-3-319-00515-7_14/TABLES/3. URL https://link.springer.com/chapter/10.1007/978-3-319-00515-7_14.

A DRAG AND DROP UI FOR USER 3D OBJECTS

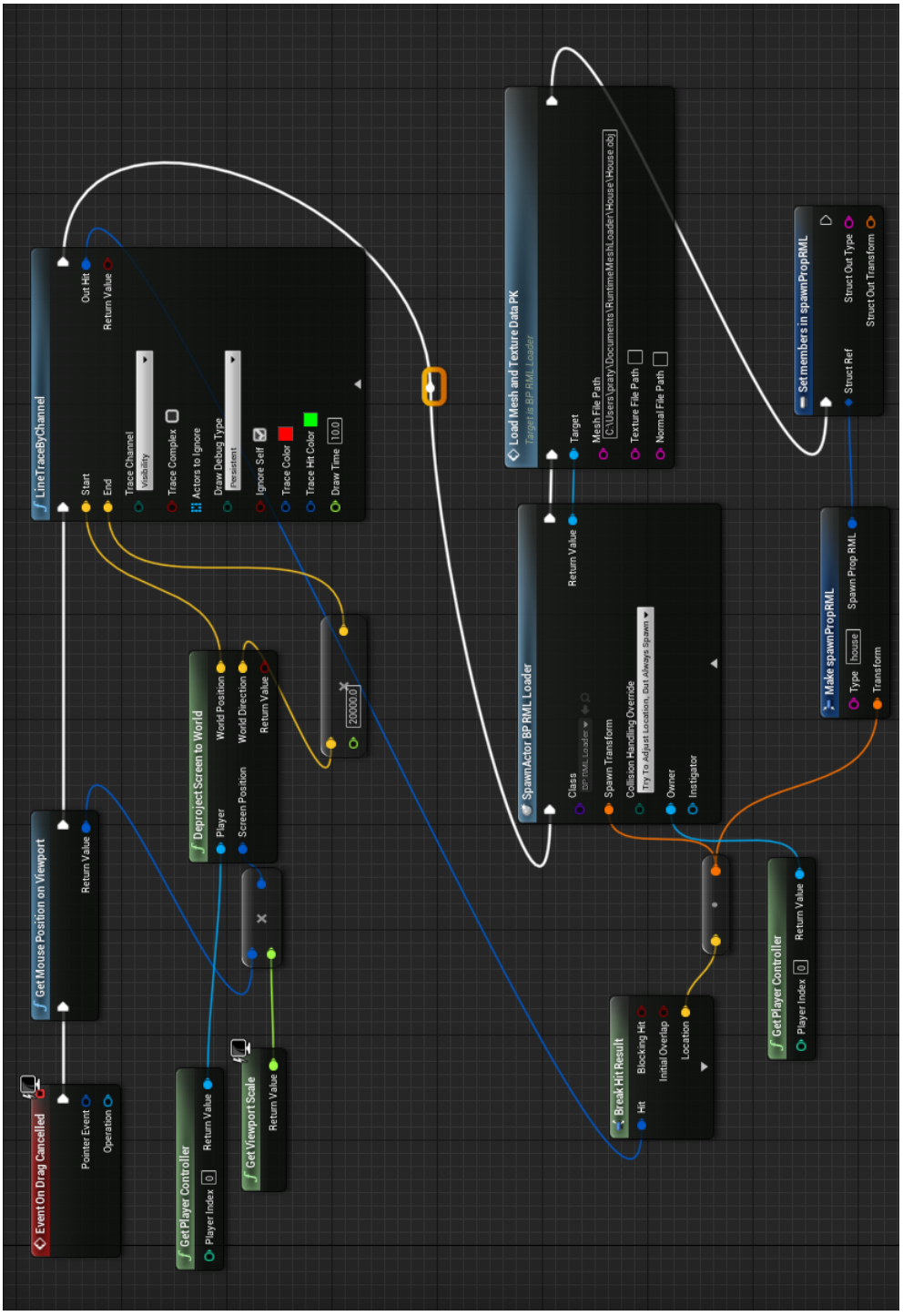


Figure A.1: Blueprint for dragging and dropping user designed 3D models

B WMS DISPLAY ON EVENT ACTIVATION

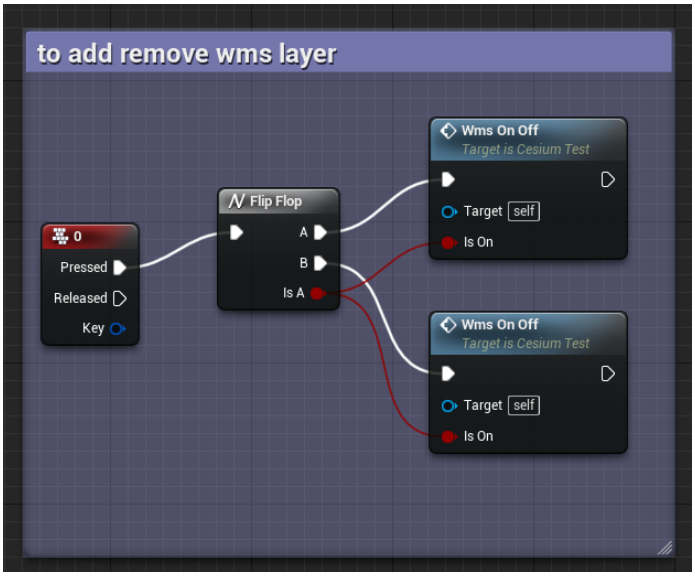


Figure B.1: Keypress event for WMS layer display in Unreal Engine

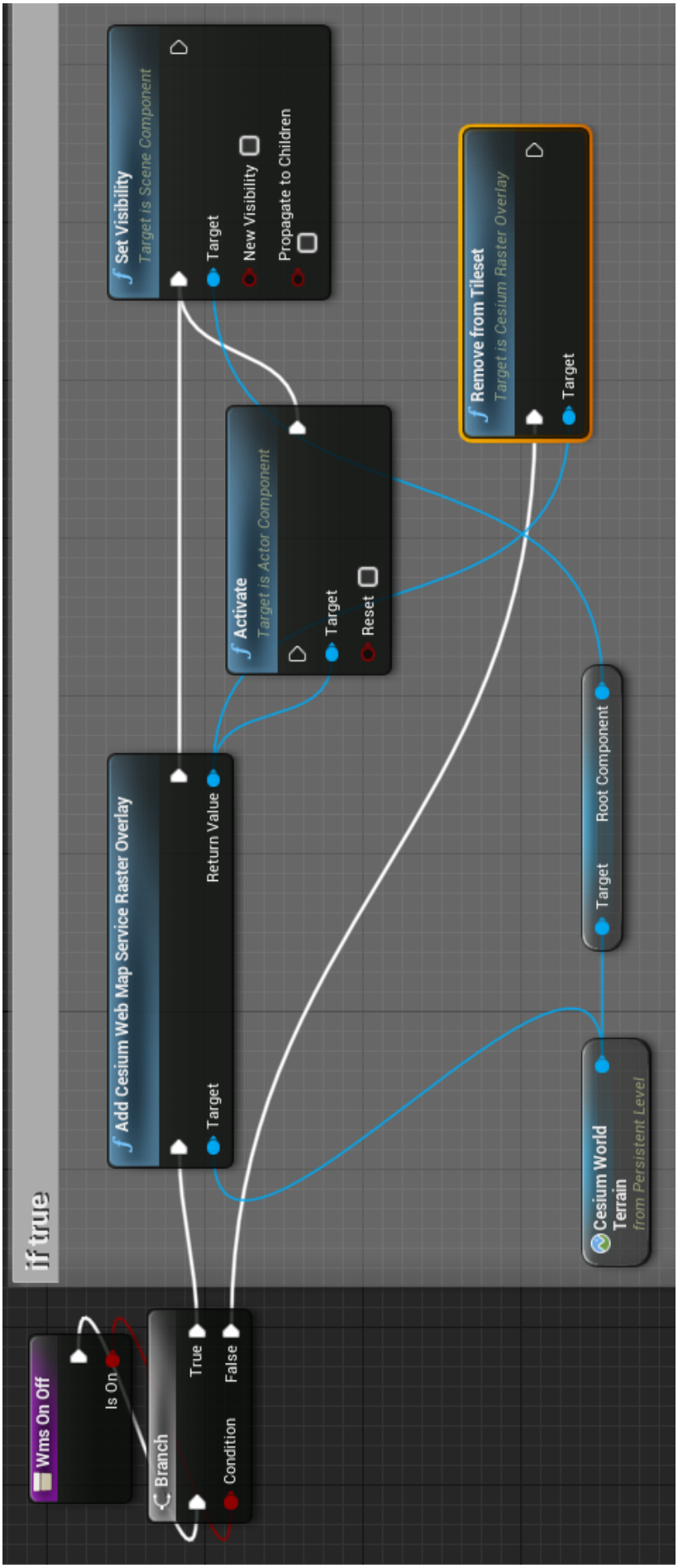


Figure B.2: Loading and unloading of WMS layer on the terrain tileset

C LOADING USER DESIGNED 3D MODELS AT RUNTIME

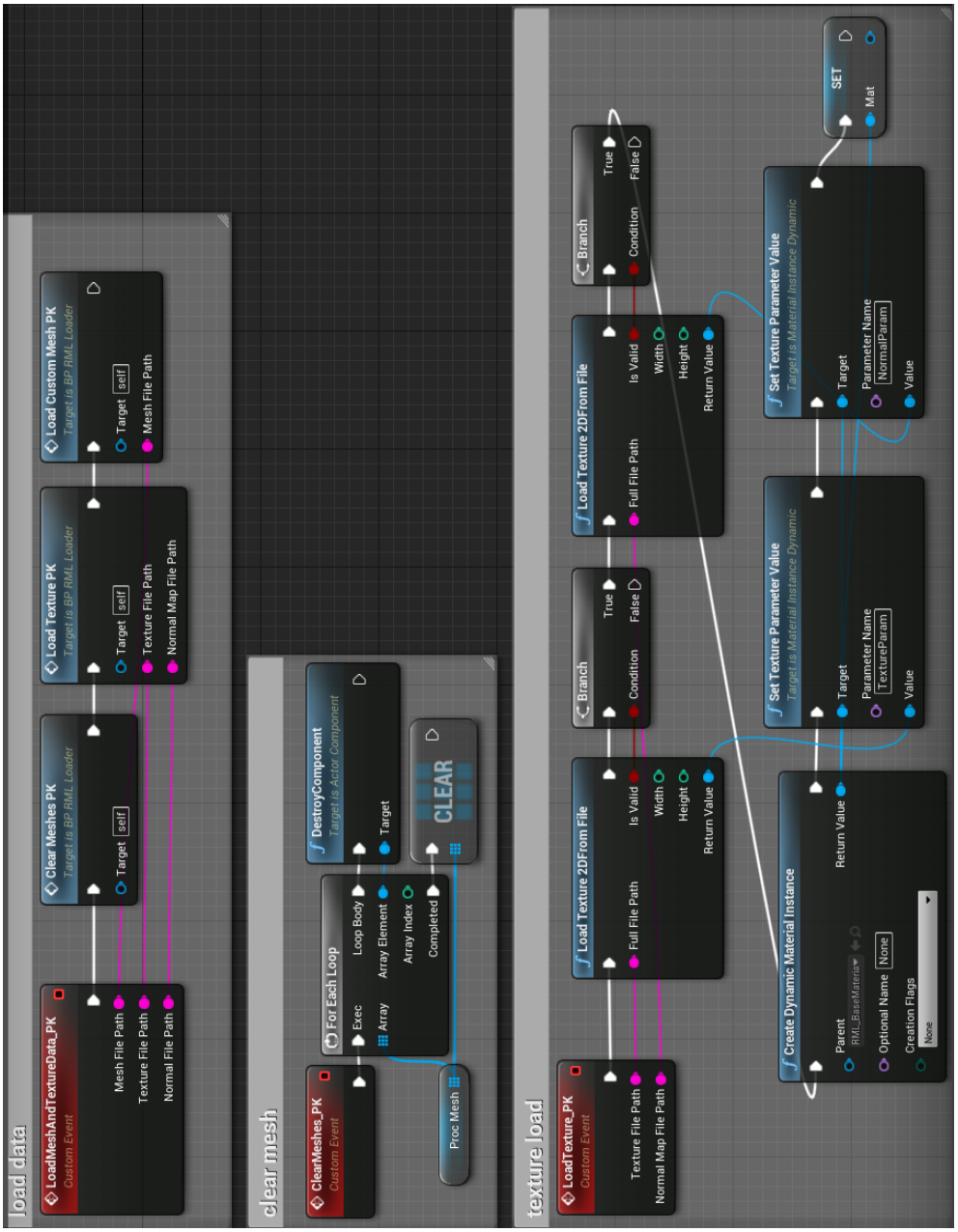


Figure C.1: Blueprint for loading model with texture

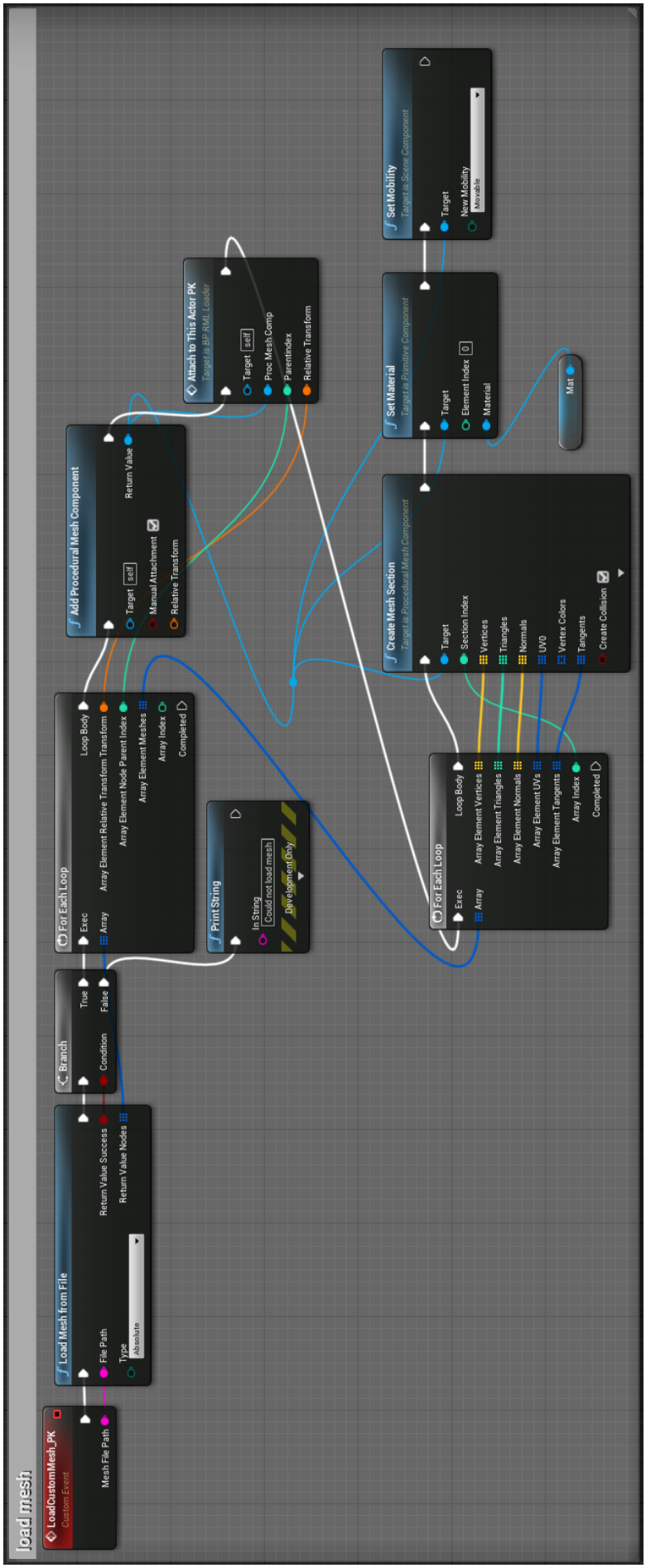


Figure C.2: Blueprint for loading model mesh

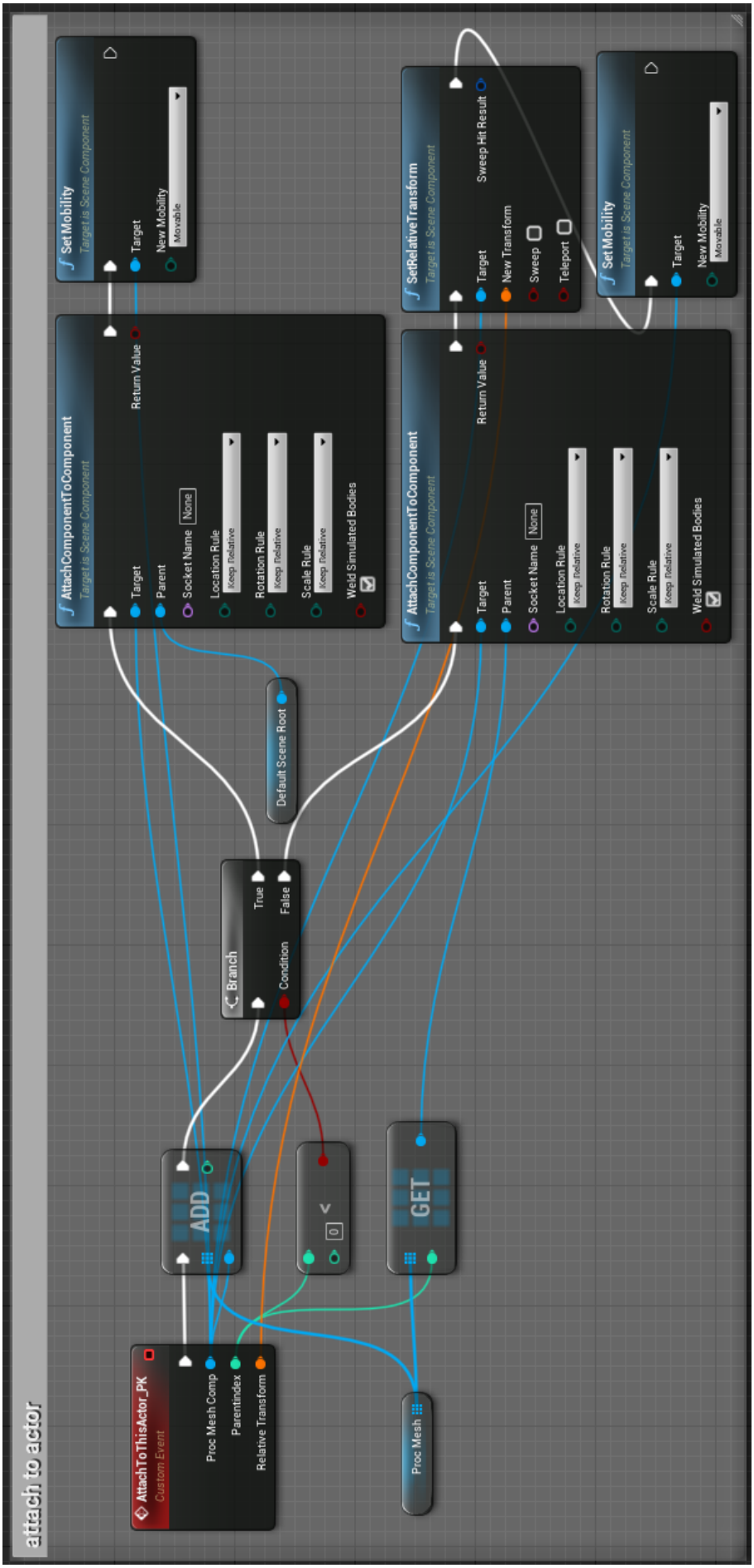


Figure C.3: Blueprint for attaching component to component

D USING GIZMO FOR TRANSFORM MANIPULATION OF SPAWNED OBJECT

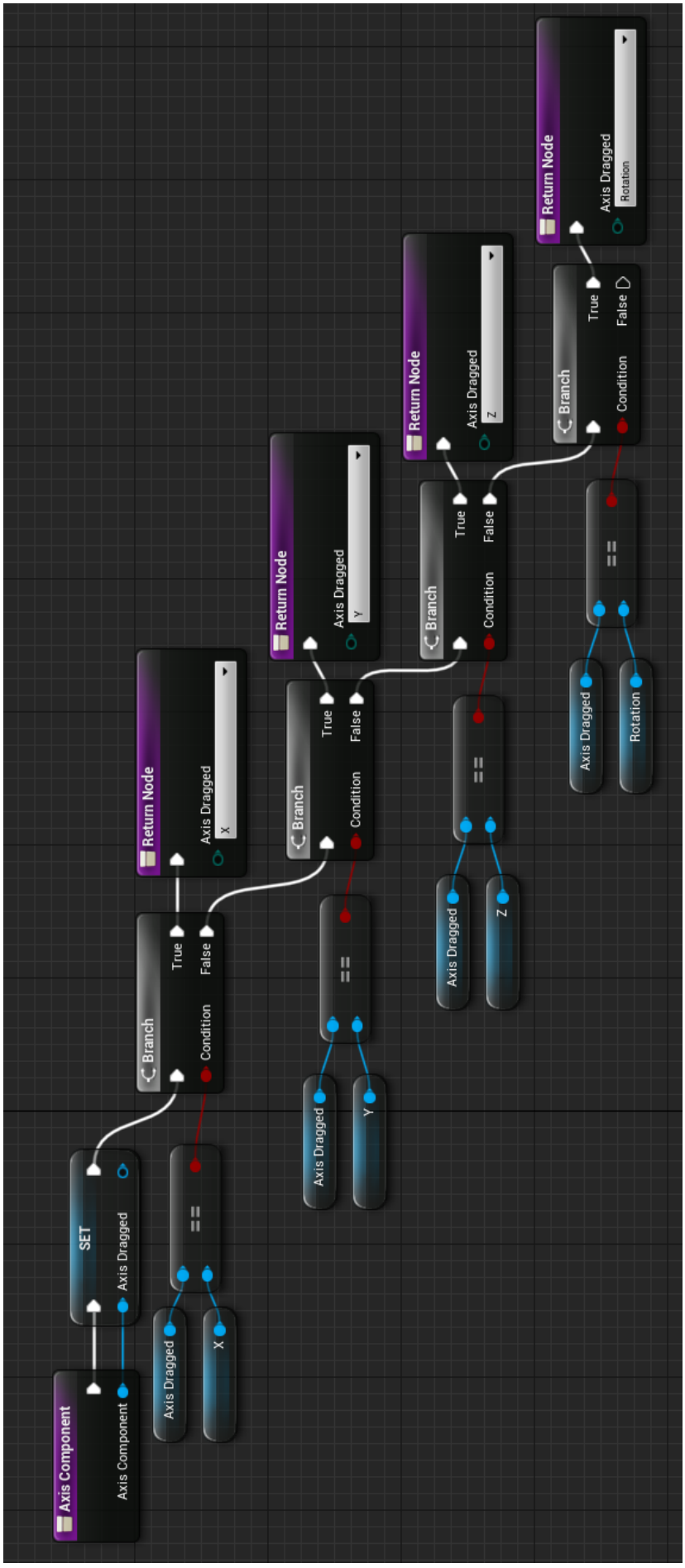


Figure D.1: Blueprint for detecting axis on gizmo

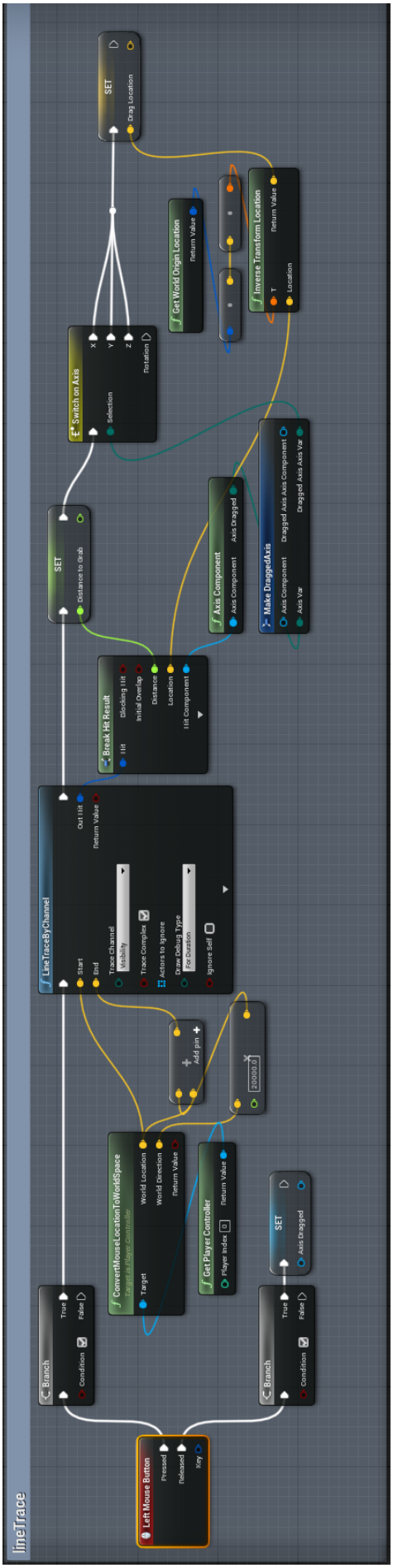


Figure D.2: Blueprint for line tracing and selection of gizmo axis

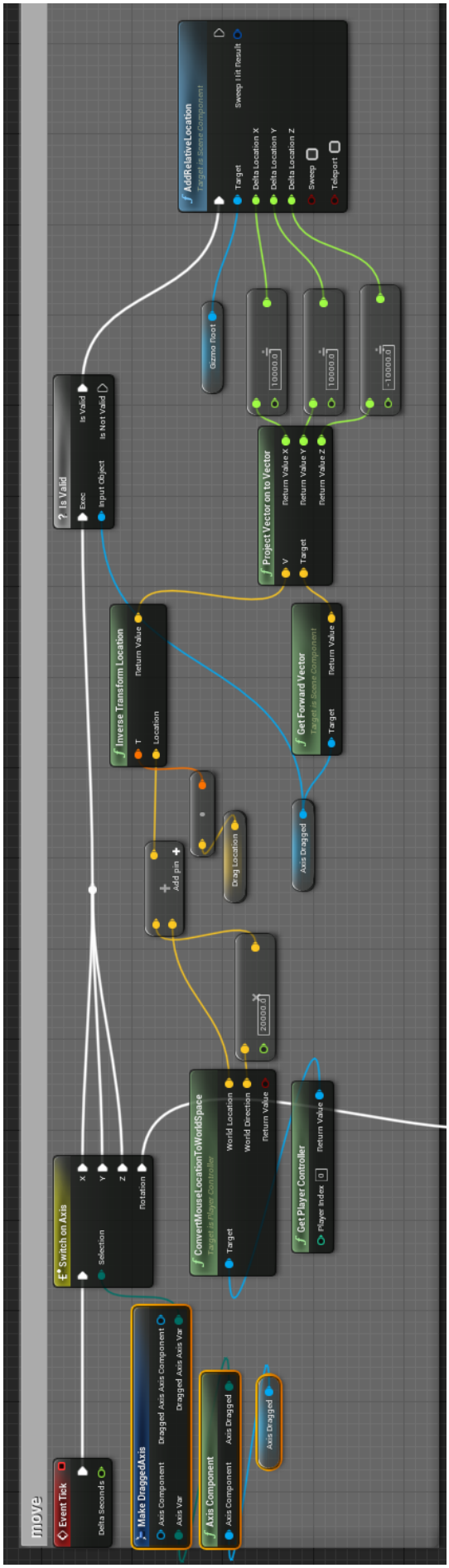


Figure D.3: Blueprint for moving object along clicked gizmo axis

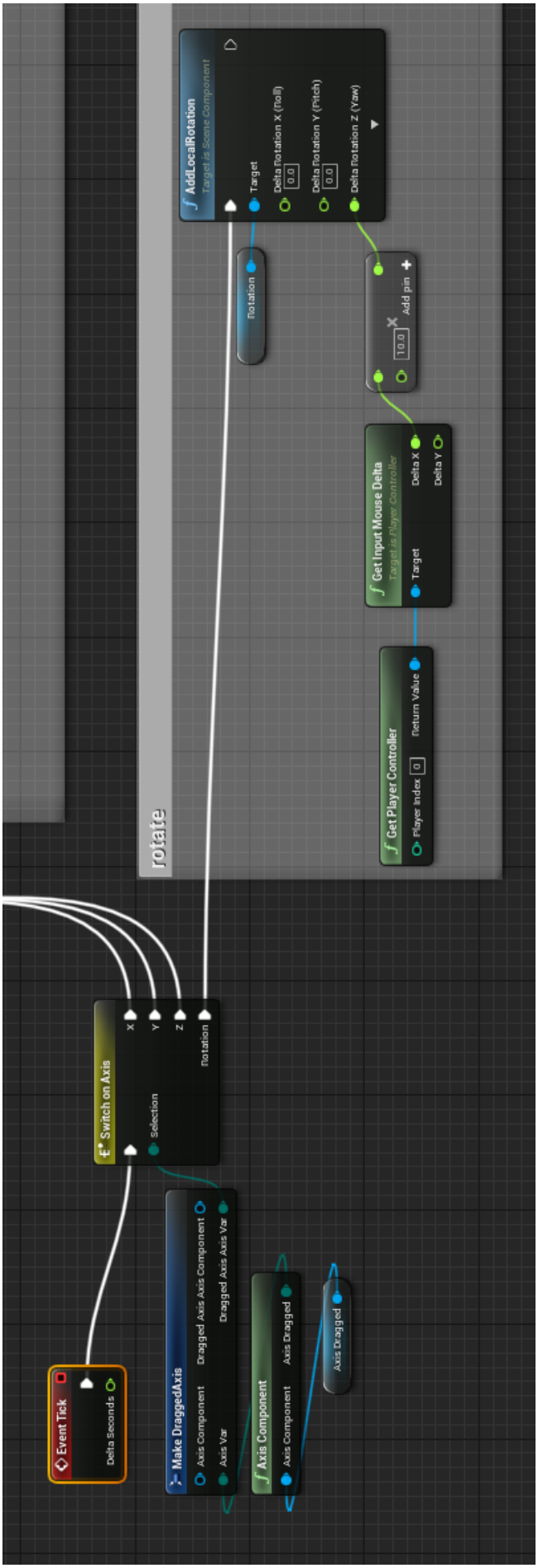


Figure D.4: Blueprint for rotation of gizmo along with spawned object

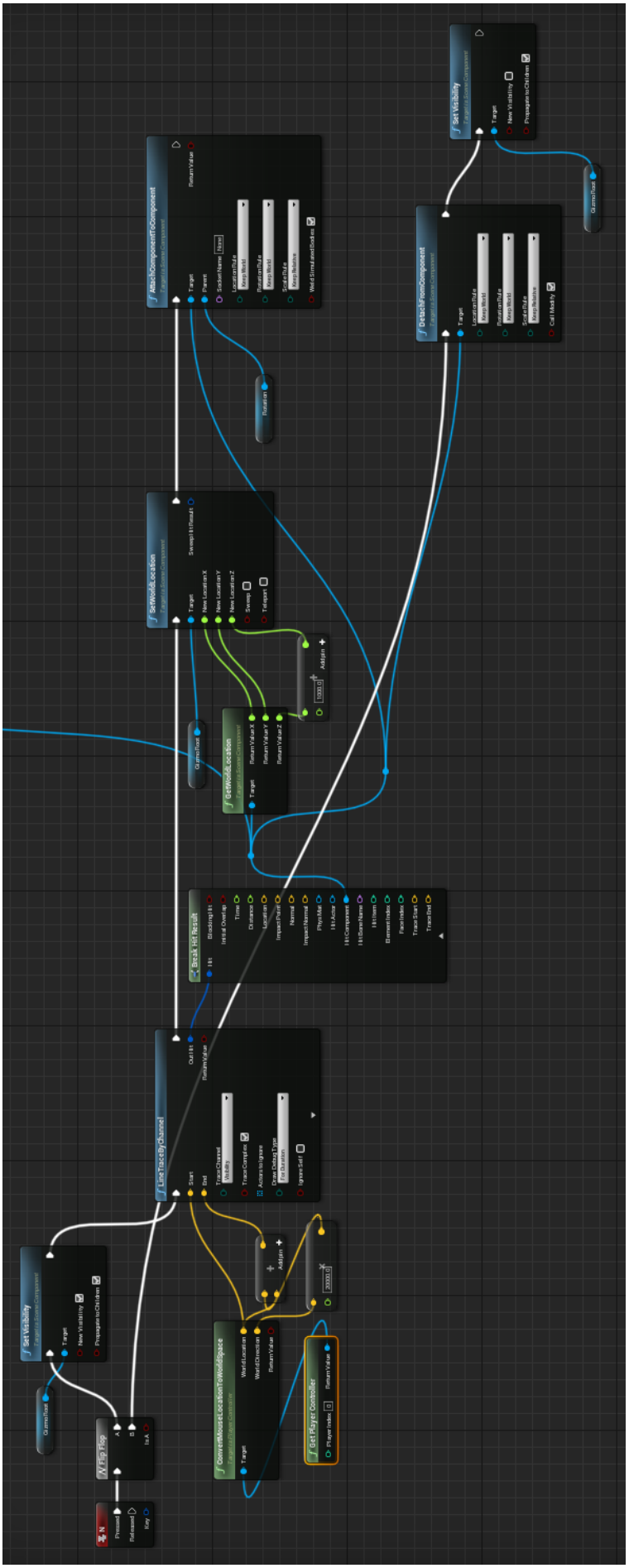


Figure D.5: Blueprint for attaching gizmo to spawned object

E UPDATING CESIUM SUN SKY TIME

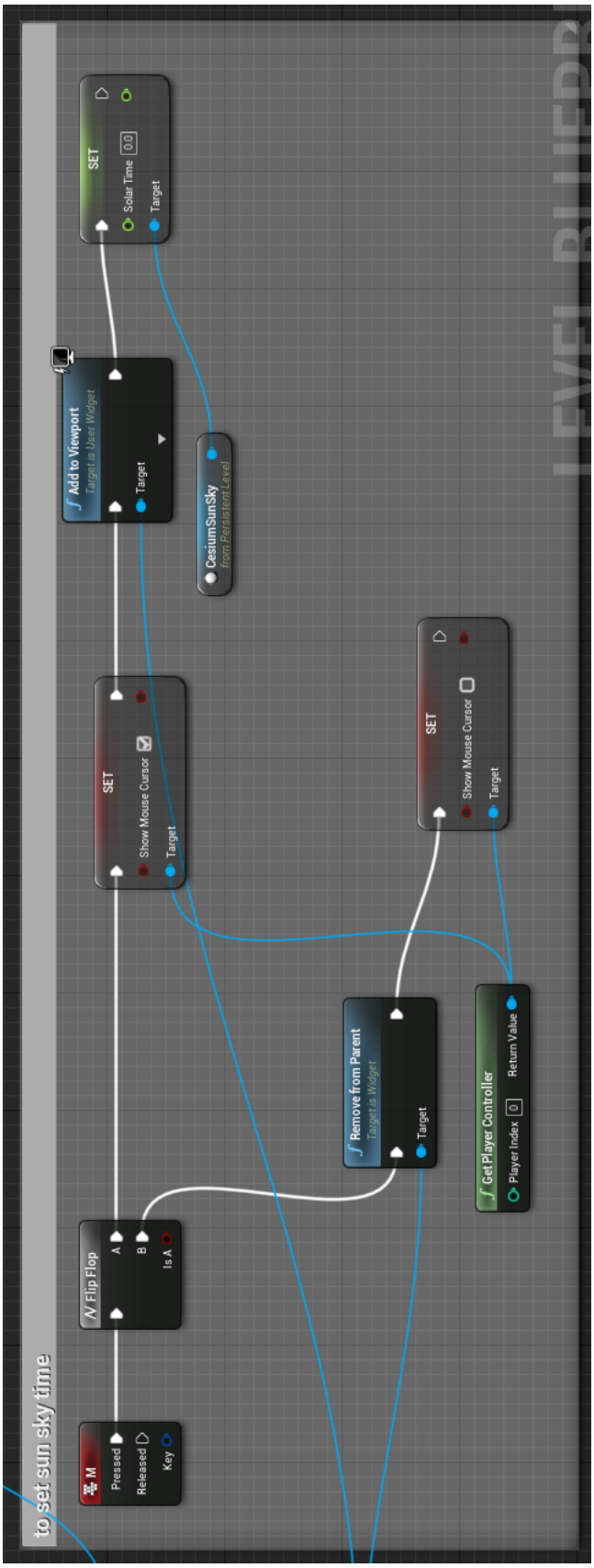


Figure E.1: Blueprint for updating Cesium sun sky using widget

F REFLECTION

This graduation thesis started officially in October 2021, and was completed within the next nine months, which stands in unison with the methodical line of approach for a thesis study done in Master Geomatics. Initial discussions and literature study done prior to the formalization of this graduation study assisted me immensely to form the research questions answered in this study. The topic was relatively new to me but I had an unwavering determination to do a research incorporating both my backgrounds in urban planning as well as geomatics and this study proves to be a good mixture of both, taking a motivation from urban planning to be able to design neighbourhoods in 3D in a VR environment, as well as including the OGC 3D tiling specification. This study taught me various skill-sets, not only technical but some which I shall continue to practice in life as well. In technical aspects, I strengthened my skills in PostgreSQL, FME, C++ and learnt a completely new software, Unreal Engine. Apart from the technical skills, I learnt a lot about proper time-management, something that I struggle with. Preparing a graduation plan and a time plan in form of a Gantt chart as well as holding meetings with my supervisors, enabled me to stay on track and stay focused with the research questions and sub-questions that were formulated.

The OGC 3D tiling technique has been used for the purpose of visualisation and dissemination of large 3D datasets, however a platform where in a combination of the 3D tiles with user-made 3D assets and designs had not been done before. This research thus, has opened opportunities for further research to be conducted in this field of geomatics, some of which have been discussed in Section 6.3.

The platformed developed as a part of this study could be used as a tool for neighbourhood design as well as an example of the capability of 3D tiles for students of geomatics. The VR capacity of the platform enables a more immersive experience and another research could be conducted on the effectiveness of using such VR based tools for the purpose of urban planning.

Learning some of the software, such as Unreal Engine, from scratch proved to be quite a daunting task and took a significant amount of time in this graduation research. It was always a work in progress and I learnt as I developed the tool and fulfilled the requirements set in the methodology chapter.

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the arsclassica package by Lorenzo Pantieri, which is an adaption of the original classicthesis package from André Miede.