Politecnico di Torino

Master of Science

Architecture Construction City

Thesis for Master Degree

GIS-Based Parametric Modeling for Architecture and Urban Digital Twins Assets

A Case Study of the Dora Park Area in Turin, Italy

Supervisors

Prof. Fabio Giulio Tonolo

Prof. Massimiliano Lo Turco

Candidate

Wang Jin

Self-Declaration of Generative AI Use

Generative AI tools were used exclusively for editorial support, such as grammar correction, translations, paraphrasing suggestions, and improving readability.

Abstract

This study explores how to combine parametric modeling and virtual reality (VR) technology with geographic information system (GIS) to improve the analytical ability and visual performance of urban digital twins. Taking the Dora Riparia and its surrounding urban parks in Turin, Italy as a case study object, this study constructs a cross-platform workflow integrating CityEngine, ArcGIS Pro and Unreal Engine 5 (UE5) through Cesium for Unreal to explore an implementable methodology that expand the range of available technical options and more broader application scenarios in the field of architectural and urban design.

The main research question examines whether Computer Generated Architecture (CGA) rule-based 3D modeling in CityEngine, when integrated with GIS-based simulation and real-time VR visualization, can deliver a more interactive and efficient urban-analysis framework compared with conventional static GIS approaches. The workflow begins with preparing vector geospatial datasets in ArcGIS Pro, which are then imported into CityEngine for parametric batch modeling, three-dimensional spatial analysis, volumetric calculations, and metric-based assessments. The detailed models produced in CityEngine are subsequently transferred back into ArcGIS Pro to support more realistic simulation tasks—such as flood-risk modelling—allowing the hydrological implications of different urban morphologies to be evaluated with greater accuracy. Thereafter, Cesium for Unreal Engine is employed to perform precise georeferencing within the WGS84 geographical coordinate, ensuring that the imported terrain, geospatial layers, and CityEngine-generated models are aligned to a consistent real-world spatial framework. This geospatial alignment enables accurate terrain matching and facilitates rendering within Unreal Engine, supporting the development of an immersive third-person VR environment.

A central result of this research is to show that when working across multiple modeling and visualization platforms, seamless flow of data is a fundamental prerequisite. In thesis, the vector data processed in ArcGIS Pro, 3D model of specific Level of Detail (LOD) parameterized in City Engine and geographic reference layer imported into UE5 can run stably only when they share the same set of coordinate system, file structure and spatial semantics. When such a unified framework is established, various types of data can be passed back and forth between different software without losing geometric accuracy, attribute information, spatial metadata, and even material mapping. The process constructed in this study shows that this data coherence is not only a technical convenience, but also directly related to the reliability and flexibility of digital twin production. Because the same data set can be read by GIS tools, parametric modeling software and real-time engine at the same time, it supports both spatial analysis and immersive visualization. This consistency allows spatial features to be presented more clearly and ensures that design discussions are always based on verifiable real geographic information. By integrating parametric modeling, spatial analysis and real-time visualization into a continuous process, this method provides a more systematic perspective for studying urban form, environmental behavior and design schemes. Although the examples in this thesis are based on specific cases, their methods can also be applied to broader fields such as smart city construction, cultural heritage recording and public participatory planning, where cross-platform data consistency is an important foundation for reliable decision-making.

In addition, the approach, methodology and overall research vision adopted in this thesis meeting the educational goals of the ARCHITECTURE CONSTRUCTION CITY Master's degree objectives[1]. The work demonstrates technical and scientific competence to interact with professionals involved in territorial transformation processes, while also reflecting the understanding of urban design and planning techniques. Finally, it further addresses the relationship between people, buildings, and the environment, emphasizing the need to relate architectural form and open spaces to human scale and use.

Contents

1. Introduction

- 1.1 Research background and significance

Over the past three decades, the rapid advancement of computer and aerospace technologies has significantly enhanced our capabilities in acquiring and processing geospatial information. Harnessing such abundant spatial information to investigate and comprehend the spatial development patterns of the Earth and human societies has gradually become a global consensus. The concept of "Digital Earth," first proposed by U.S. Vice President Al Gore[2], articulated the urgent desire to develop and apply the already acquired geospatial information. A year later, the International Society for Digital Earth was established at the joint initiative of scientists from Canada, the United States, China, Japan, the Czech Republic, and other nations[3]. Its mission is to promote international academic exchanges and project collaborations, and to facilitate the application of "Digital Earth" technologies in crucial areas such as national economic and social sustainable development, environmental protection, disaster management, preservation of world heritage and natural resources, as well as counter-terrorism and the maintenance of global stability. Owing to its strategic significance in agricultural development, urban management, resource exploration, environmental protection, disaster prevention and mitigation, crime combat, and peacekeeping, Digital Earth has emerged as one of the globally focused research hotspots since the inaugural International Conference on Digital Earth. In alignment with this trajectory, the European Commission has launched the Destination Earth initiative, a flagship program aimed at building a highly accurate digital twin of the Earth system in order to monitor, simulate and predict interactions between natural phenomena and human activities, thereby enabling informed adaptation strategies and supporting resilient environmental governance[4].

With the evolution of digital city technologies, applications primarily relying on 2D data can no longer meet the demands of various professional urban applications. A more intuitive, what-you-see-is-what-you-get form of 3D spatial data has gradually emerged as a novel and user-favored means of data expression, set to become the core data for both digital cities and the broader Digital Earth initiative. Consequently, urban 3D modeling, as a vital method for generating 3D spatial data, has garnered significant attention. It constitutes an indispensable component in constructing the foundational framework of a digital city. By converting real-world architectural elements and other features into three-dimensional geometric models that can be visualized and rendered on a computer screen, it enables the most intuitive and convenient human-computer interaction with geographic information. The appropriateness about data acquisition including sensors, platforms and technologies as well as modeling method directly influences the quality of 3D model outputs and even the evaluation of the entire digital city framework. Modeling serves as the cornerstone of 3D visualization and analysis; most basic geographic information platform software for digital cities operates on models to facilitate essential operations such as navigation, observation, analysis, and decision-making.

Representing and visualizing large urban areas has long been a demanding task in computer graphics. The difficulty stems not only from the sheer quantity of geometric information that a city contains, but also from the heterogeneity of the datasets used to describe it. Urban form reflects

layered historical, cultural, social, and economic processes, and this complexity translates into a landscape with high functional diversity and considerable visual variation.

With improvements in computing performance and the development of more efficient modeling tools, city-scale environmental models can now be processed on standard desktop computers. In this broader context, urban modelling can be understood from two complementary perspectives: geometric modelling and systems modelling. Geometric modeling focuses on the physical representation of the built environment-including buildings, terrain, streets and infrastructure-emphasizing spatial precision and visual consistency (Figure 1). In contrast, system modeling focuses on the dynamic behaviors that shape urban life, such as traffic flow patterns, energy consumption, and environmental processes.

Representing and visualizing large urban areas has been a challenging task in computer graphics. The difficulty lies not only in the huge amount of geometric information contained in cities. It also lies in the heterogeneity of the data set used to describe the city. The urban form reflects historical, cultural, social and economic processes, this complexity translates into Landscape with high functional diversity and significant visual changes.
Improved computing performance and development of modeling tools. It is possible to deal with city-scale environments on computers. In this broader context, urban modeling can be accomplished through two Complementary perspectives to understand: geometric modeling and system modeling. Geometric Modeling. The concern is to give the built environment (buildings, terrain, streets and Infrastructure) in physical form, giving priority to spatial accuracy and visual consistency (Figure 1). On the other hand, system modeling focuses on dynamic behaviors that shape urban life, such as traffic patterns, energy consumption, and environmental processes.
When these two dimensions are combined, the urban model can gain deeper explanatory power.
Programmed 3D modeling is no longer limited to the generation of scenes with excellent visual effects, it has become a means of exploring, assessing and predicting the state of cities. This integrated approach enables the modeling workflow to be applied not only to visualization practices, but also to analytical tasks and scenario-based design in urban planning and virtual environment development.

When these two dimensions are brought together, urban models gain explanatory depth. Programmatic 3D modeling is no longer limited to producing visually appealing scenes; it becomes a means to explore, evaluate, and anticipate urban conditions. This integrated approach allows modeling workflows to contribute not only to visualization practice but also to analytical tasks and scenario-based design within urban planning and virtual environment development. (Figure 2).

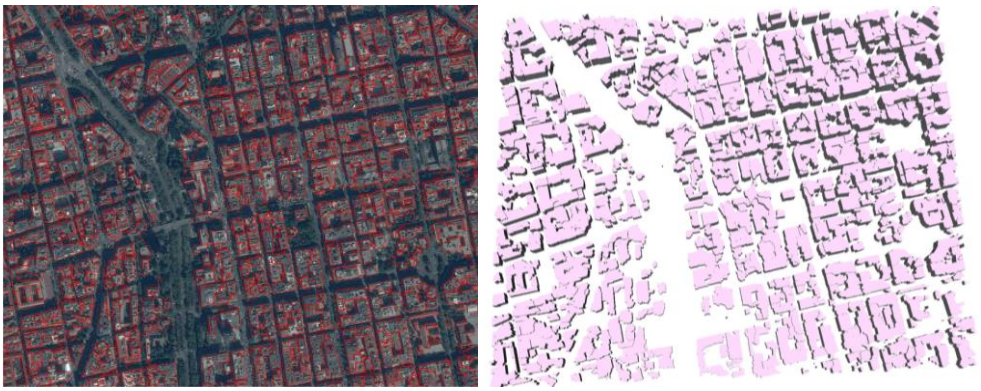


Figure 1. Reconstruction of Madrid downtown from satellite imagery
Source: *https://inria.hal.science/tel-02275865v1*

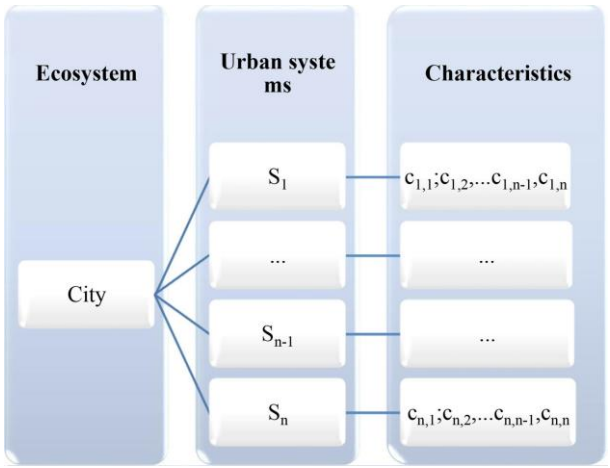


Figure 2. The system modeling employs a multi-level hierarchical framework, encompassing different urban systems (S1, S2, … Sn), with the corresponding constituent features (c1, 1, … cn, n) of each system being divided at different levels. Based on this framework, decomposing the city into its constituent features and their interdependencies is crucial.
Source: *https://doi.org/10.1016/j.cacint.2024.100166*

In terms of geometric model establishment, conventional approaches to 3D modeling for digital cities, such as 3D MAX and SketchUp, are labor-intensive manual processes that consume substantial time and effort, with post-modification proving cumbersome. In recent years, a new modeling technique—parametric modeling—has gradually come to the forefront. Parametric technology regulates and constrains the form of designed objects through a set of parameters, where explicit correlations exist between parameters and morphological characteristics (Figure 3). Modifications to design outcomes are driven in real-time and dynamically by these parameters. Modeling systems developed using parametric technology can fully leverage the powerful computing capabilities of computers to rapidly generate urban road and building models in accordance with predefined parameter conditions and constraint rules. Moreover, adjusting parameters allows for immediate generation of different results without duplicating workloads.

Figure 3. Metropol Parasol in Seville, Spain. By Architects J Mayer H
Source: *https://www.dezeen.com/2011/04/26/metropol-parasol-by-j-mayer-h/*

Building on its research team's expertise in parametric urban modeling, parametric architectural modeling, parametric reconstruction of ancient buildings, interactive road modeling, and urban simulation, the Swiss company Parametric developed its software CityEngine in 2008. It designed a shape grammar (Shape Grammar) underpinned by parametric technology, tailored for 3D urban scene modeling, to define a series of grammatical rules. These rules drive 2D planes or 3D models to continuously generate greater details, thereby creating a wide range of complex 3D models[5]. The software has found extensive application in game development, digital scene creation for films, urban planning, architectural design, and the preservation of ancient buildings.

Parametric technology is poised to bring revolutionary breakthroughs to the fields of architectural design and urban planning across theoretical, methodological, and technical dimensions. Currently, numerous renowned institutions like Zaha Hadid Architects, SOMA Architects and College of Architecture in Tongji University specializing in architecture and urban planning have identified parametric technology as a cutting-edge research direction and have established related parametric design laboratories. It is an indisputable fact that parametric technology has become a major research focus and important development trend in architectural design and urban planning.

In another aspect of system modeling, models are often used to characterize the types of mobility and dependency structures that affect urban operations, such as traffic cycles, energy demand, or how flood risks propagate. By combining parametric city models with analytical data sets, users can assess how geometric changes affect system behavior. For example, associating the regularized model of CityEngine with ArcGIS Pro allows for a scenario analysis that includes land density, accessibility, and environmental impact. Similarly, when these data are further linked with the real-time visualization function of Unreal Engine, planners can explore the dynamic characteristics of urban systems in an interactive way, thus promoting a more data-driven and predictable urban design workflow.

Unlike geometric modeling—which focuses on generating form and achieving visual accuracy—systems modeling brings time, behavior, and changing conditions into the digital-twin workflow. By layering these dynamic components onto the physical model, the analytical scope of the digital

environment expands, making it possible to explore how urban areas evolve under different social or environmental pressures. When the two modeling approaches are used together, they create a link between spatial representation and functional performance, shifting the model from mere visual depiction toward a more process-oriented understanding of the urban landscape[6].

The area selected for this study is Dora Park and its surrounding urban environment in northern Turin, Italy (Figure 4). The site was chosen because it provides a relatively balanced intersection between the natural environment and the built-up space. Near to the Dora Riparia River, this area shows the continuous interaction between the city's ecosystem and urban renewal. The river corridor not only shapes the topography of the site, but also affects the spatial distribution of surrounding green space, residential areas and public infrastructure.

Figure 4. The case area of interest in the thesis

In addition to its environmental qualities, the area includes several architectural and cultural landmarks, such as the Chiesa del Santo Volto, which mark the ongoing transformation from former industrial zones to mixed-use urban spaces. The presence of parks, pedestrian routes, and residential blocks creates a diverse setting that captures different urban morphologies within a manageable scale (Figure 5). These characteristics make Dora Park particularly suitable as a case study for spatial modeling and visualization, as it provides both realistic complexity and accessible data for testing digital twin workflows.

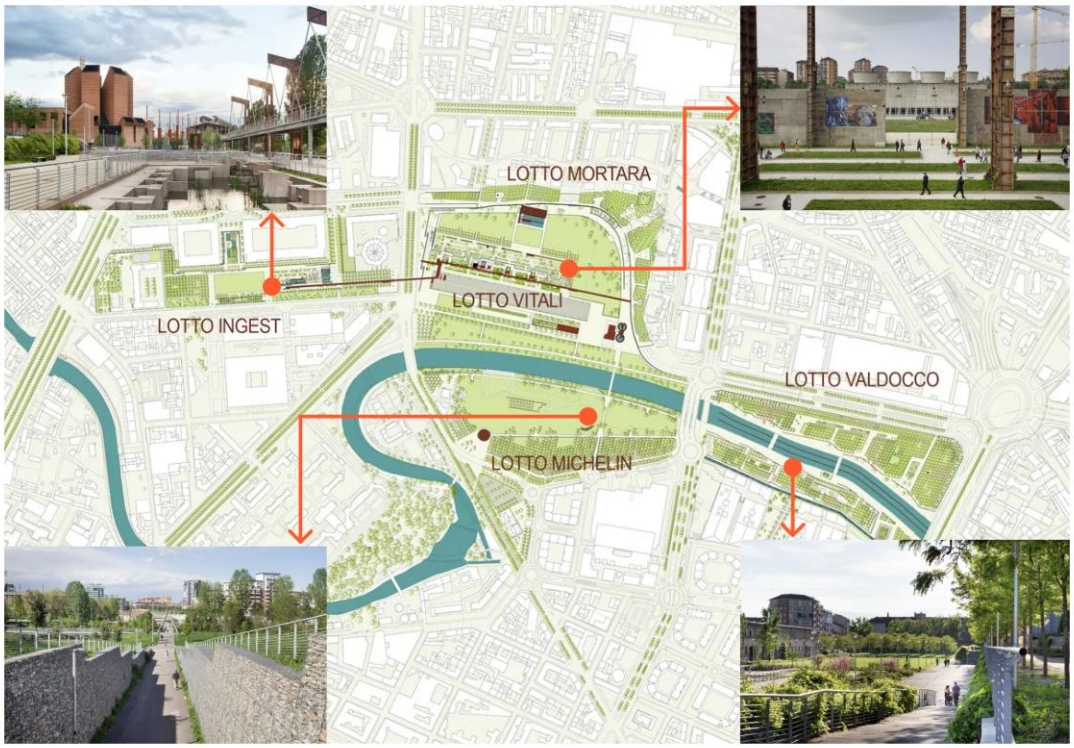


Figure 5. Doral Park Industrial site transformation and renovation by Latz + Partner
Source: *https://www.latzundpartner.de/en/projekte/postindustrielle-landschaften/parco-dora-turin-it/*

- 1.2 Research content and methods

This thesis explores the technical process of integrating parametric modeling techniques with GIS data to support digital urban modeling and visualization. The study focuses on testing the feasibility and workflow efficiency of these technologies. The methods rely primarily on ArcGIS Pro, CityEngine, and Unreal Engine (Cesium), forming a complete digital chain from data acquisition and model generation to immersive visualization. The research process supports the architecture and urban construction, emphasizing digital representation, spatial analysis, and the use of computational tools in design practice.

Research Significance:
1. GIS data integration
GIS spatial data constitutes the starting point of the whole modeling process. When topo surfaces, road networks and building outlines are imported into CityEngine according to their original coordinate reference system (such as WGS 84 / UTM), they can be consistent with real-world locations. This spatial consistency enhances the credibility of the three-dimensional environment and demonstrates how geographic information can enhance the authenticity and analytical value of architectural and urban models.
2. Technology Exploration
In this study, CGA based parameterization rules are used in City Engine to realize the automatic

generation of building and road networks. This process is used as a practical verification of rule-based modeling method, which proves that it can reduce repeated manual operations and improve the overall modeling efficiency. By repeatedly adjusting the attributes and rule parameters, the generated geometric information can be closer to the actual urban form.

3. Virtual Reality Visualization

The models generated in CityEngine are then imported into Cesium for Unreal Engine and accurately matched to global terrain data, resulting in accurate alignment spatially. In this way, the study area can be visualized immersivity in a virtual reality environment, which helps to understand the spatial structure more clearly and improve the effect of result expression and design communication.

Research Methods:

1. Acquire cartographic data and orthophotos from regional geospatial platforms (e.g., Geoportale Piemonte: https://geoportale.igr.piemonte.it/).
2. Process the datasets in ArcGIS Pro, ensuring that coordinate reference systems are unified before 3D modeling.
3. Import the GIS data into CityEngine and apply CGA rules to generate buildings, streets, and terrain components.
4. Adjust rule attributes and parameters so that the resulting geometries align with real-world measurements and spatial characteristics.
5. Use the generated models in CityEngine for visualization, measurement, and street-network analysis. The interoperability of the workflow also allows these models to be transferred into ArcGIS Pro for three-dimensional flood-simulation studies.
6. Export the models to Cesium for Unreal Engine, align them with the WGS84 coordinate system, and visualize them as an interactive urban scene.

- 1.3 Definitions (Digital twin, Parametric modeling, Virtual Reality)

Digital Twin:

The term Digital Twin was coined by “Michael Grives” in his article “Virtually Perfect: Driving Innovative and Lean Products through Product Lifecycle Management”[7] and which setting a foundation for the developments of Digital Twins, digital Twin is mainly composed of three distinct parts: physical Part in the Real world, virtual Part in the Virtual space and data connection that ties the physical and virtual parts together. Grieves conceptualized this interaction as a continuous cycle linking the physical and virtual domains: data flows from the physical entity to its virtual counterpart, while information and processes are transferred back from the virtual space to the physical system. The virtual domain may in turn be composed of multiple specialized sub-spaces, each designed to support distinct functions such as modelling, testing, and optimization (Figure 6). The Digital Twin is also generally understood as comprising three core components: a physical asset, its virtual representation, and the data channels that integrate the two. Increasingly, it is being investigated as a mechanism for enhancing the performance of physical systems by exploiting computational methods made possible through their digital counterparts.

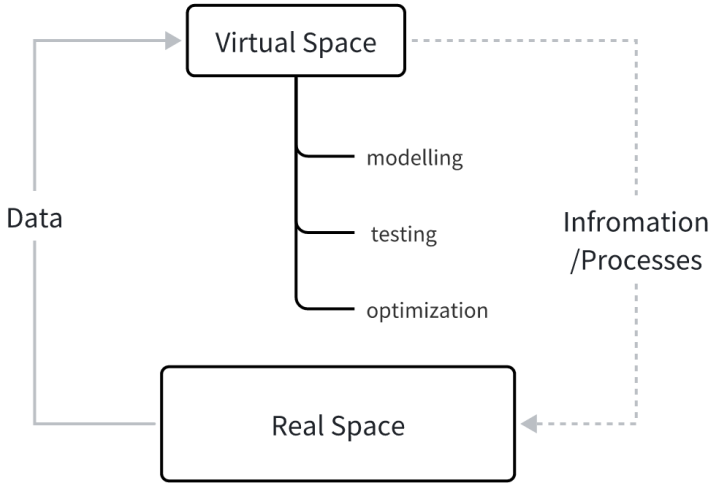


Figure 6. Mirroring or Twinning between the physical and virtual spaces.

Research on the subject has grown significantly in recent years. Although the number of studies varies by industry, the overall trend is continuing to rise. As shown in Table 1, according to the statistics of Scopus, Science Direct and Google Scholar (data retrieved in May 2022), the number of publications related to Digital Twin (DT) technology has increased rapidly since about 2016, marking an accelerated development stage in this field. In specific application areas, manufacturing still dominates, followed by energy, construction and education. Although the application of DT in the field of education started late, the attention of academic circles has increased significantly in recent years[8].

Industry	No. of Publications		
	Scopus	Science Direct	Google Scholar
Aerospace and Aeronautics	168	980	2360
Manufacturing	1823	3295	25,300
Healthcare and Medicine	188	819	4810
Power Generation/Energy	780	2863	17,300
Automotive	177	939	7910
Oil and Gas	434	1463	9780
Smart City	236	462	4090
Mining	191	1001	9700
Maritime and Shipping	59	360	2330
Agricultural	63	495	3900
Education	223	939	14,900
Construction	748	1825	17,700
Retail	21	286	2830

Table 1. Distribution of Digital Twin research publications across different industries and databases. Source: Maulshree Singh et al., *Applications of Digital Twin across Industries: A Review*. 2022.

Despite this momentum, the literature still lacks a unified and stable definition of the Digital Twin, as well as a clear perspective on how the concept continues to evolve to address its expanding range of applications. This absence of consensus has resulted in a proliferation of characterizations and terminologies surrounding digital twins and the twinning process. Consequently, the wide

variation in frameworks adopted across different sectors risks fragmenting the concept and undermining its intended benefits[9].

Michael Grieves later expanded the Digital Twin framework by linking it to the broader product life cycle and defining four interconnected components: the Digital Twin Prototype, Digital Twin Instance, Digital Twin Aggregate, and the Digital Twin Environment (Table 2). Within the life-cycle structure (Figure 7), the Twin first appears as a Prototype during the design stage. When the product enters the realization phase and physical units are manufactured, corresponding Instances are created. The collection of all such Instances constitutes the Aggregate. Both the Individual Instances and the Aggregate operate within the Digital Twin Environment, which represents the virtual counterpart of the product's real operating context and supports activities such as modelling, simulation, and performance assessment. Notably, these Instances, along with the Aggregate and the Environment, may remain active even after the physical product has been retired or removed from service[10].

Concept	**Description**
Digital Twin	A complete virtual description of a physical product that is accurate to both micro and macro level.
Digital Twin Prototype	A virtual description of the prototype product that contains all the information needed to create its physical twin.
Digital Twin Instance	A specific instance of a physical product that remains linked to an individual product throughout that product's life.
Digital Twin Aggregate	The combination of all the Digital Twin Instance.
Digital Twin Environment	A multiple domain physics application space for operating on Digital twins. These operations include performance prediction, and information interrogation.

Table 2: The list and descriptions of key concepts surrounding the Digital Twin.

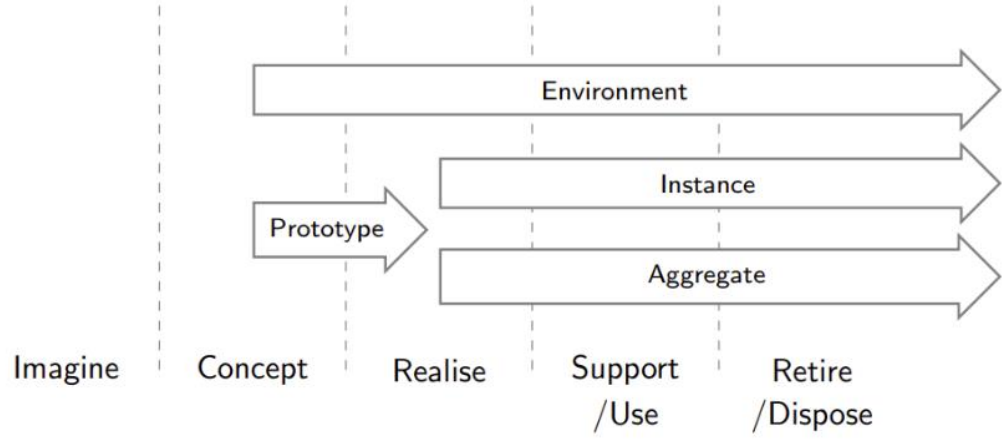


Figure 7. The scope and transitions/relationships between the Digital Twin elements and physical product.
Source: David Jones et al., *Characterising the Digital Twin: A systematic literature review.*2020.

At its foundation, the Digital Twin concept describes a framework that links a physical asset with a corresponding digital model so that information can circulate between the two. Data generated during the operation of the physical object are collected and examined, and the resulting insights support not only the improvement of the existing product but also the development of subsequent iterations. In Grieves's formulation, this creates a knowledge-based, data-driven system capable of observing, managing, and optimizing an asset throughout its entire life cycle.

Parametric modeling:
Parametric modeling is a design strategy that relies on parameters, constraints and algorithmic rules to generate geometric shapes rather than drawing shapes one by one. In this method, the designer sets the relationship between elements, so that the adjustment of a small number of key variables can automatically reshape the overall model. This approach builds a highly flexible structural system that allows a large number of design alternatives to be tested and compared quickly and consistently.
In the field of architecture, parametric modeling marks a shift from fixed and isolated geometry to a systematic design paradigm composed of interrelated components. By embedding dependencies and procedural rules in the model, local modifications will affect the overall composition, making the design process more adaptable and responsive [11].

As Kolarević (2003) [12] notes, this method constructs a framework connecting early conceptual design with later construction implementation, so that digital process can penetrate design intention into actual construction. By encoding geometric logic and material constraints directly inside the model, parametric systems can transform design ideas into buildable forms with minimal information loss.

Schumacher (2009) [13] extended this discussion to a broader paradigm in architecture, proposing that parametrium is not only a set of technical tools, but also a cultural mechanism to promote the development of design logic, which can strike a balance between formal diversity and overall consistency. The Kartal-Pendik master plan of Zaha Hadid Architects (Figure 8) is a typical case. The project aims to build a new urban sub-center in Istanbul Asia to ease the pressure on the historic city. The project site was originally industrial land, adjacent to the fine urban texture of surrounding suburbs.

According to the principle that parametric design emphasizes avoiding abrupt spatial transformation, the surrounding context, especially the traffic flow line of the merging site, becomes the key basis for shaping the overall spatial structure. By using Maya's hair dynamics tools, these paths are parametrically aggregated into wider road corridors to delimit larger development plots and form a horizontal street network whose logic echoes Frei Otto's minimum detour principle. A major north-south urban axis defines the overall orientation, while a series of parallel secondary streets reinforce the overall structural framework. The resulting spatial pattern combines the characteristics of a minimal detour network with a slightly deformed mesh.

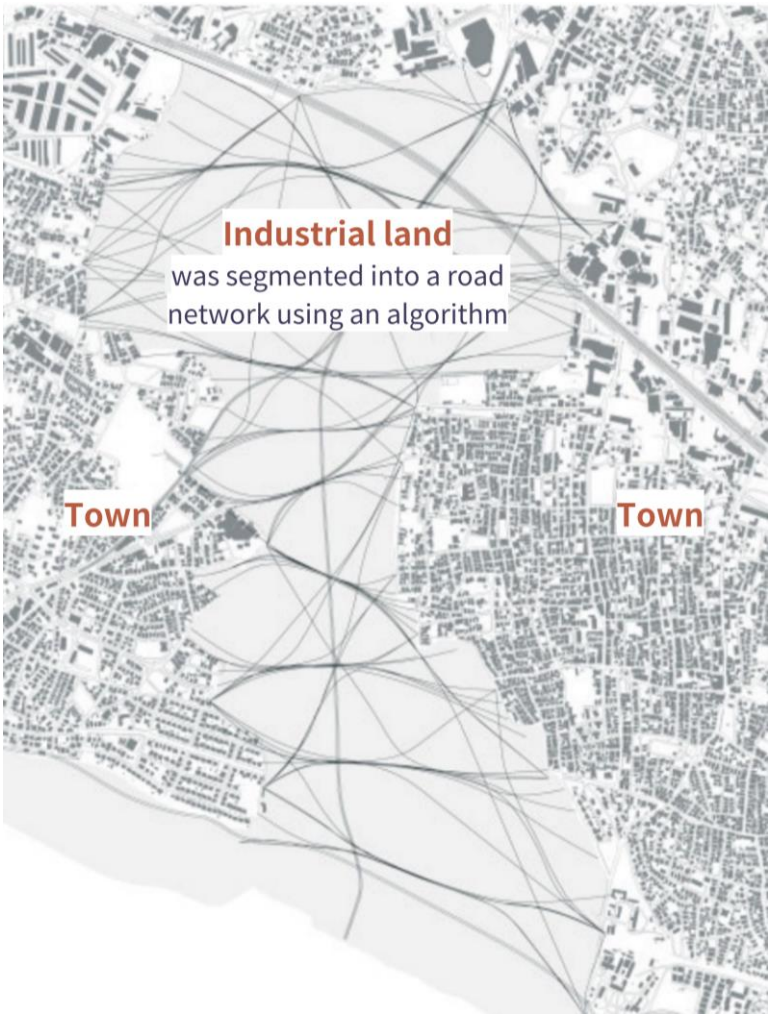


Figure 8. The minimal-detour network was produced through Maya's hair-dynamics engine, where circulation paths were generated using a digital thread-based simulation workflow.
Source: *https://www.zaha-hadid.com/masterplans/kartal-pendik-masterplan/*

Contemporary research has shown the relevance of parametric modeling in sustainability, optimization, and digital twin development. Caetano (2020) [14] , for instance, highlights its capacity to integrate structural, environmental, and functional data into multi-objective frameworks, making design more responsive to complex performance criteria.
In the study by Imke Drave et al., a Digital Twin of dam infrastructure was constructed based on a Building Information Modeling (BIM) framework[15]. The model was designed to connect with existing systems, such as integrating Cyber-Physical Systems (CPS) with Information Systems, or linking the Digital Twin with Internet of Things (IoT) networks. During operation, annotations or supplementary models can be utilized to merge models from different perspectives into a Decision Tree (DT) (Figure 9). This integration connects the models with sensor data derived from intelligent civil structures, thereby incorporating data visualization directly into the model for real-time monitoring and analysis.

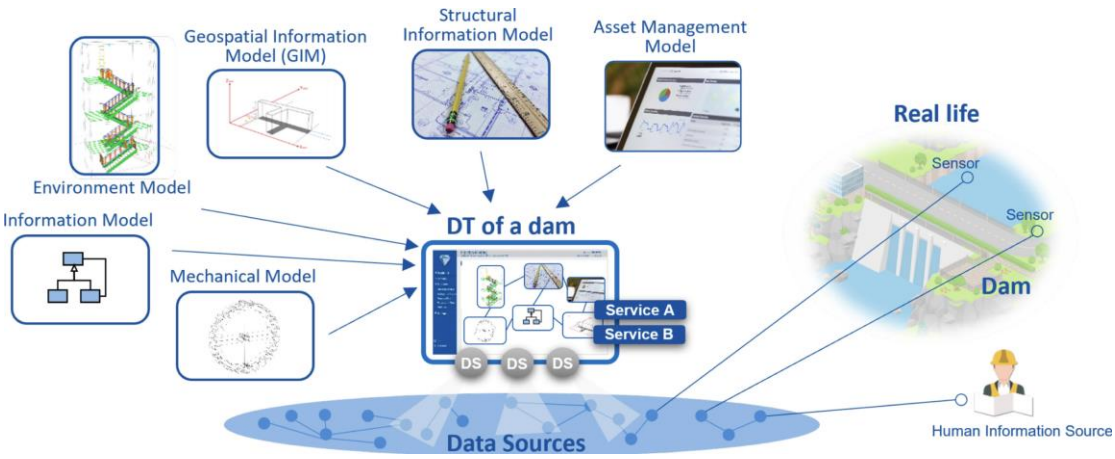


Figure 9. Integration of parametric modeling perspectives for intelligent infrastructure in digital twin applications.
Source: Judith Michael et al., *Integrating models of civil structures in digital twins: State-of-the-Art and challenges.* 2024.

Taken together, parametric modeling can be defined as a computational paradigm where rules, parameters, and constraints guide design generation. It empowers architects and engineers to maintain adaptability, pursue performance-driven solutions, and strengthen the connection between digital conception and physical realization.

Virtual reality:
Augmented Reality (AR) and Virtual Reality (VR) represent two distinct yet related technologies that integrate digital information with human spatial perception. AR overlays virtual information onto the physical world in real time. Rather than replacing reality, it enhances it by adding digital layers through mobile devices or AR glasses. AR is thus more suitable for design validation and on-site decision support, whereas VR enables full-scale immersion and spatial experience.
As a technological tool for constructing immersive digital environments, Virtual Reality (VR) has sparked extensive discussions—its core value lies in enabling users to perceive a sense of "being present" and interact with computer-generated content. Among the existing definitions of VR, the framework proposed by Steuer (1992) [16] has exerted a profound influence: it identifies vividness and interactivity as two key dimensions that determine telepresence, where telepresence refers to the psychological experience of "existing in a mediated environment." This theoretical framework remains essential for analyzing applications like Virtual Reality Tours, given that such tours depend on the combined effects of immersion and interaction to replicate the experience of spatial exploration.

The traditional concept of a "virtual tour" refers to a digital simulation of real-world locations, usually constructed through panoramic images, video clips, or 360-degree photographic materials. While this form of virtual tour allows users to browse target locations remotely, it suffers from obvious limitations in delivering immersive experiences and supporting real-time interaction (El-Said, 2021) [17]. In comparison, a Virtual Reality Tour (VR Tour) expands on this traditional concept by integrating VR hardware devices—including headsets and motion controllers—and interactive three-dimensional models. This integration effectively enhances users' spatial perception abilities and

strengthens their sense of presence. Recent research in the fields of tourism and cultural heritage has explicitly acknowledged this distinction, defining VR tours as immersive and interactive experiences that enable users to explore destinations in a way similar to physical on-site visits.

Empirical research shows that VR virtual tour can significantly affect user experience and shape their subsequent behavior. For example, Ouerghemmi et al. (2023) [18] f pointed out that carefully designed virtual tours can enhance users' sense of presence and improve their ability to form a mental image of the destination, thus improving their willingness to actually go to the place.
These findings suggest that the value of VR virtual tours extends far beyond the travel industry. For areas such as architecture, cultural heritage preservation and digital twinning - all of which rely on immersive, highly invested visualization to convey spatial quality and enhance user perception - VR technology is also important.

2. Scientific background

- 2.1 Digital twins (city)

In the field of urban construction, the digital twin of a city or building based on 3D modeling is a system of interconnected digital twins, representing certain aspects of the functioning and development of the urban environment. These digital twins support fine-tuning and synchronization with the real state of urban infrastructure through data from various sources in real-time.

Various aspects, such as housing, infrastructure, transportation, and energy are intertwined in a city, and urban problems occur in the complex interrelationships of each[19]. Therefore, an urban digital twin applied with digital twin technology in an urban space is a comprehensive technology system that cannot be composed of simple technology or a single application, and is considered a key element and starting point of a smart city[20].

In the urban digital twin, the actual urban space is reproduced in the same digital space; in the process, all the domains and systems of the city are reflected in this digital platform. In this virtual city, real-time interaction between physical reality and the virtual model takes place[21], including real-time monitoring of the city, analysis of various phenomena, prediction of the future through simulation, and visualization of various characteristics[19]. Through this interaction, policymakers can quickly and accurately understand the current state of the city at a low cost, and have the opportunity to predict the effect of the policy and respond to the situation before implementing the policy.

One example is the city of Lörrach on the border with Switzerland. Lörrach provides its citizens with a public 3D city model using CESIUM (3D geospatial platform allowing accurate integration of terrain, buildings on a virtual globe) in which designs and plans for building objects are visualized in the context of the city's surroundings like Figure 10. This makes it possible to see at an early stage how, for example, lines of sight change and plans fit into the surrounding built environment.

Figure 10a. Central Hospital planning area draft in the 3D city model of the city of Lörrach (City of Lörrach)

Source: *https://loerrach.virtualcitymap.de/#/*

Figure 10b. Human viewpoint of Central Hospital in the 3D city model of the city of Lörrach (City of Lörrach)
Source: *https://loerrach.virtualcitymap.de/#/*

- 2.2 Parametric modeling

Today, computational and statistical analyses play a significant role in architecture and urban planning, especially with the evolving methodologies in urban planning processes[22]. The adoption of cutting-edge technologies has become essential for the design and execution of urban planning projects. Parametric modeling, in particular, facilitates the generation of multiple design variations or scenarios by establishing architectural element variables and defining the interrelationships among them[23].

Parametric models are based on mathematical probability and statistical principles. This allows users to adjust the value of one variable to observe its effect on other variables. This capability enables designers to generate new design scenarios without having to redraw the model each time, as shown in Figure 11. Essentially, parametric modeling enables designers to make modifications to the overall form rather than just adjusting individual components.

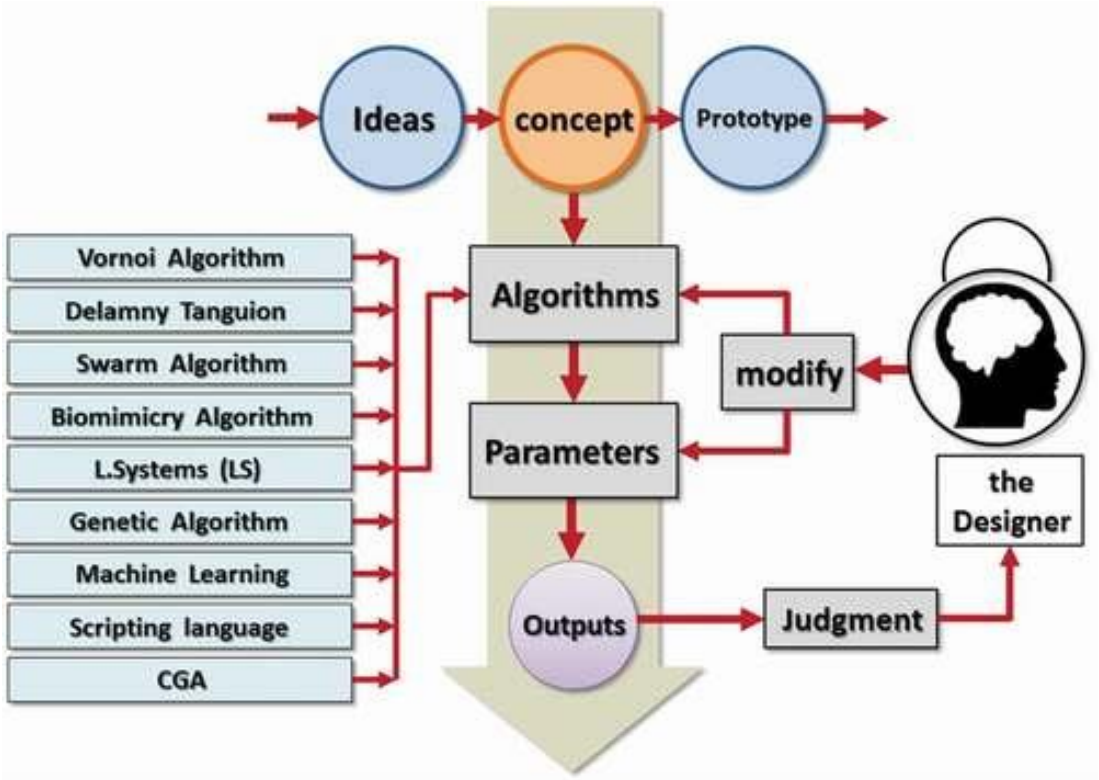


Figure 11. Framework of Parametric Models.
Source: Ibrahim M. Badwi et al., *3D-GIS Parametric Modelling for Virtual Urban Simulation Using CityEngine*. 2022.

At present, parametric modeling methods have been widely embedded in a variety of mature software systems, including CATIA (Computer Aided 3D Interactive Application), APOLLON, SolidWorks. In the field of architecture, relatively main platforms are Grasshopper in Rhino, BIM tools (such as Autodesk Revit) and CityEngine used in this thesis.

Different platforms have different advantages, in the field of architectural and city, although they are all called parametric modeling, they have different construction logic and application scenarios from platforms.
Grasshopper in Rhino is particularly effective for generating functionally controlled architectural forms through algorithmic design, especially when dealing with complex geometries or smooth, elegant curves. It is also suitable for urban-scale simulation tasks such as shortest-path analysis and solar exposure studies.
In contrast, BIM platforms focus primarily on project management and detailed construction information at the building scale. Meanwhile, rule-based parametric modeling in CityEngine operates at the city and regional scales, emphasizing the relationships between architecture, urban form, and the environment.

CityEngine is a specific platform in the Urban Parametric Model (UPM), which is a computer-based framework that represents the various components of an urban system by translating them into a set of variables that govern the composition of urban elements in three dimensions—such as buildings, roads, sidewalks, lighting poles, trees, shades, and seating. The parametric design

approach offers a methodology for defining these design elements as adjustable variables[24]. Users can modify the initial values of these variables and incorporate predefined rules and criteria, allowing for the generation of numerous alternatives for a single building footprint. This approach enables the restoration of the urban context's vision and the creation of an infinite number of design scenarios.

Parametric design tools are particularly advantageous because they keep the design open to modifications while maintaining a high level of detail, facilitating a deeper understanding of the essential qualities and elements of the proposed design. For instance, as shown in Figure 12, MVRDV has begun construction on "Valley" (formerly known as P15 Ravel Plaza) — a 75,000 m^2 mixed-use complex in Amsterdam's Zuidas business district that integrates public leisure areas, offices, and residences. The project adopts a stepped green-terraced architectural form whose shape was generated through parametric design, ensuring that sunlight reaches all 196 apartments within the complex, each featuring a unique layout.

The outline of each floor is not drawn line by line, but generated by preset rules and parameters, such as lighting conditions, visual field requirements and structural constraints. The parametric tool automatically verifies that the design of each room meets the appropriate performance criteria, including solar heat gain, structural limitations, and sunlight requirements.

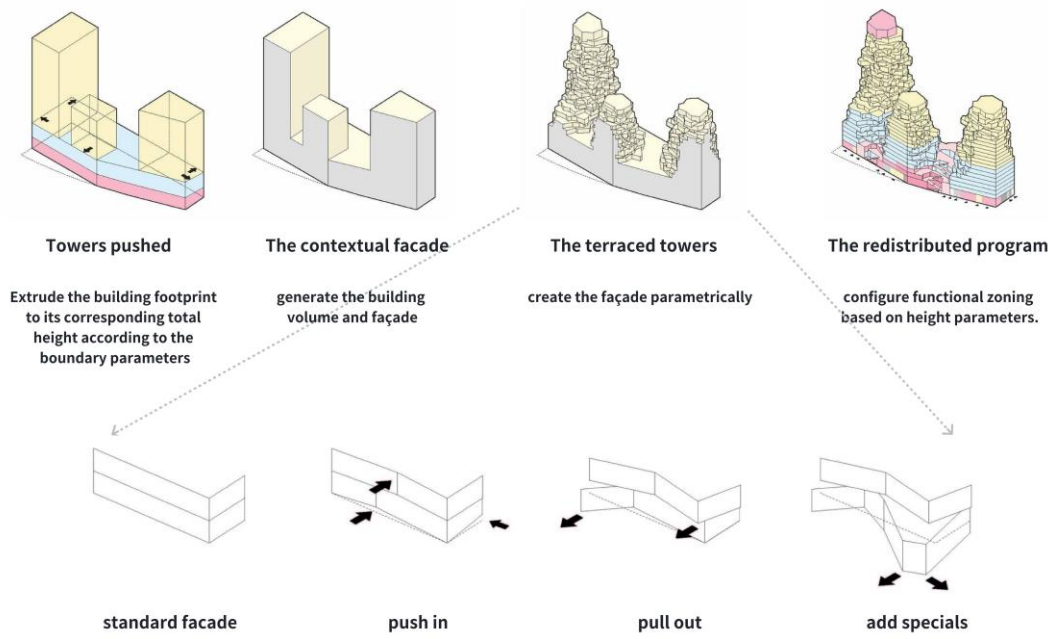


Figure 12. The general process of parametric-assisted modeling and optimization.
Source: *https://www.archdaily.cn/cn/879184/*

- 2.3 Virtual reality (VR) tour and interactivity

Virtual Reality (VR) Tour in the context of game engines refers to an immersive, interactive experience created using game development software where users can explore a virtual environment as if they were physically present. These tours can be of real or fictional spaces, allowing users to navigate freely or follow guided paths within a 3D environment. Game engines like Unity and Unreal Engine are often used to create these tours, providing advanced graphics, physics, and interactivity to enhance the experience.

There are many precedents for virtual reality tour technology.

Matterport 3D is a platform and technology that specializes in creating immersive 3D virtual tours of real-world spaces. It combines 3D scanning, photography, and cloud-based processing to generate highly detailed, interactive models of physical environments, which can be explored online in a VR-like experience. In Matterport 3D virtual tours like Figure 13, which provides 3D Virtual Tours, Measurement Tools, Real Estate and Marketing and Facilities Management and Construction features.

Figure 13. Pompei Experience Captured with Matterport Pro2 (Analist Group)
Source: *https://discover.matterport.com/search?q=Pompei*

Benoit Dereau is a freelance CG generalist creating Archi-viz scenes using Video-Game Realtime

Engine who created the Unreal Paris Apartment using Unreal Engine 4 in Figure 14, It showcases the engine's ability to render high-quality environments that users can explore in VR. In this VR tour, users can navigate the apartment freely, interacting with elements like lights, furniture, and doors, making it an immersive way to experience architectural spaces before they are physically constructed.

Figure 14. Folio of Unreal Paris 2018
Source: *https://www.unrealengine.com/es-ES/blog/building-unreal-paris*

These precedents demonstrate how VR tours and interactivity can be applied across various fields, from cultural heritage and education to real estate and automotive marketing, providing users with rich, immersive experiences that go beyond traditional media.

3. Related platform and their relationships

- 3.1 ArcGIS pro (ESRI)

- 3.1.1 Software Overview

ArcGIS Pro[25] is a desktop-based GIS software developed by the Environmental Systems Research Institute (ESRI), USA. As the core desktop product within ESRI's geospatial technology ecosystem, it is designed to provide a high-precision, integrated technical support platform for Geographic Information Science (GIScience) research, spatial data analysis, and industry-specific applications. Serving as an upgraded and iterative replacement for the traditional ArcGIS Desktop (ArcMap), ArcGIS Pro centers its positioning on "end-to-end spatial data management and in-depth analysis." It integrates modern computer graphics, spatial database technology, and industry application logic, and while retaining the data collection, processing, visualization, and output capabilities of traditional GIS software, it further emerges as a critical technical vehicle connecting spatial data, analytical models, and decision-making applications through its modular architecture and cross-platform collaboration features.

From a technical point of view, ArcGIS Pro is an integrated platform for processing and analyzing spatial information. Its core advantage is to solve the problem of incompatible spatial data formats. It also makes up for the fragmentation of work that is common in traditional analytical processes.
Compared to open source GIS programs such as QGIS, which typically focus more on basic spatial operations or domain-specific functionality. The design concept of ArcGIS Pro is to provide a complete spatial information service process.
The software supports a complete analysis chain: from importing and standardizing raw data (such as satellite images, GNSS observation points and vector layers), to integrating multi-source data through raster-vector fusion or two-dimensional and three-dimensional data association, and then to implementing more advanced spatial analysis methods, including network analysis, spatial interpolation and geographically weighted regression. In addition, it provides rich visualization features such as dynamic graphics, 3D scene construction, and interactive dashboards.
The geographic database structure (File Geodatabase and Enterprise Geodatabase) adopted by ArcGIS Pro can store massive data efficiently. It also enables multiple people to access this data at the same time. At the same time, Python scripting capabilities and Model Builder automate repetitive tasks. They also standardize complex analytical processes. Together, these capabilities provide a solid technical platform for interdisciplinary research. Application fields include environmental science, urban planning, public health and other directions.

In terms of application positioning, ArcGIS Pro possesses the dual attributes of being an "academic research tool" and a "carrier of industry solutions." For researchers in the field of GIS science, its high-precision spatial analysis modules (e.g., Spatial Analyst, 3D Analyst, Network Analyst extensions) can support the verification of complex scientific questions—such as simulating regional air quality distribution via spatial interpolation models or optimizing public transportation route planning

through network analysis. For practical application scenarios involving government departments and enterprises, ArcGIS Pro can achieve spatial data sharing and rapid output of decision-making results through collaboration with ESRI's cloud platforms (e.g., ArcGIS Online, ArcGIS Enterprise). For instance, urban planning departments can build urban land expansion models based on this software, generate visual planning scheme drawings, and synchronize them to the cloud for collaborative review across multiple departments; traffic management departments can dynamically optimize traffic control strategies by combining real-time traffic data with spatial analysis models.

ArcGIS Pro's technical characteristics are also reflected in its in-depth support for "3D spatial information expression and analysis." Different from the planar presentation of spatial information in traditional 2D GIS software, ArcGIS Pro is equipped with a built-in 3D Scene module, which enables integrated modeling and interactive analysis of multi-dimensional spatial data (e.g., terrain surfaces, building models, underground pipelines). In urban disaster prevention research, for example, the coupling of 3D urban models and flood inundation simulation models can be used to intuitively present urban areas affected by different flood levels, providing more reality-aligned technical support for disaster prevention decision-making. Furthermore, its compatibility with Open Geospatial Consortium (OGC) standards (e.g., support for WMS and WFS geoservice calls) allows for data interoperability with other GIS or industry-specific software (e.g., BIM software Revit, remote sensing processing software ENVI), further expanding the value boundary of spatial information in cross-domain applications.

In summary, ArcGIS Pro is not only a " spatial data processing tool", but also a " digital decision support system with spatial information as the core". Through technology integration, process standardization and application scalability, it builds an effective technical bridge between GIS scientific research and industry practice. For academic research, it provides a highly accurate and repeatable spatial analysis method. For practical application scenarios, it provides a visual and intelligent spatial decision solution. Therefore, ArcGIS Pro has become an indispensable core technology platform in the field of contemporary spatial information.

- 3.1.2 Advantages in Urban 3D Modeling of ArcGIS pro

ArcGIS Pro is particularly beneficial in urban 3D modeling, which is crucial for effective urban planning and management:

1. Scenario Planning and Simulation:
Urban planners often rely on ArcGIS Pro to explore alternative development options and to assess how these choices might influence the built environment. By testing multiple scenarios, they are able to make more grounded decisions on zoning regulations, transport systems, and future infrastructure investments.
A practical example comes from a study on flood risk in Austin, conducted by researchers at the University of Texas. In their work, ArcGIS Pro was used to assemble a three-dimensional flood impact scene that combined flood-depth raster data, an urban digital terrain model, and detailed asset layers. Bringing these datasets together allowed the team to visualize how water would spread across the city and which infrastructures might be exposed during severe events. The resulting 3D

environment provided a clear analytical basis for discussions on urban resilience and emergency-management strategies. (Figure 15).

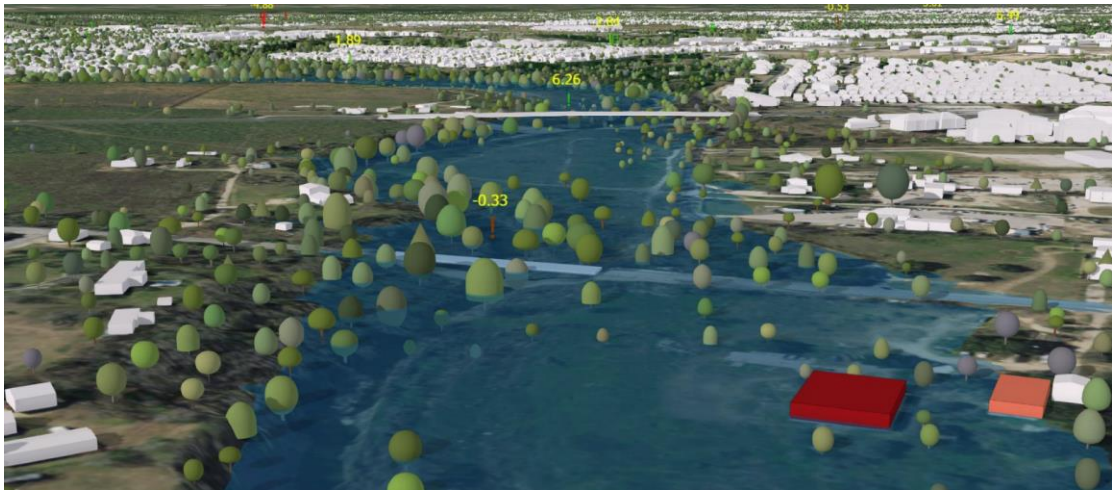


Figure 15. Austin, Hurricane Harvey impact analysis in ArcGIS Pro
Source:*https://www.arcgis.com/home/webscene/viewer.html?webscene=3455af9be32141569da41f42d880d022*

2. Integration with Real-World Data:

ArcGIS Pro supports the integration of high-accuracy 2D and 3D metric datasets such as LiDAR and satellite imagery, oblique photography height measurement, into 3D models. This enhances the accuracy and realism of the models, leading to more reliable planning and analysis.

For example, in recent research, Song et al. (2018) demonstrated an integrated approach that combines ArcGIS Pro–based spatial analysis with Digital Surface Models (DSM) to assess rooftop solar energy potential. In their study of Beijing's Chaoyang District, two-dimensional rooftop outlines were extracted from high-resolution remote-sensing imagery, while DSM data were used to capture elevation and slope variations, enabling the accurate differentiation of building structures from surrounding terrain (Figure 16). Within the ArcGIS Pro analytical framework, the DSM supported the computation of three-dimensional roof parameters—such as slope, aspect, and typology—which were subsequently used to simulate solar radiation and photovoltaic efficiency.

This method demonstrates that the combination of GIS data and DSM elevation information can significantly improve the geometric accuracy of 3D urban models and enhance the reliability of subsequent analysis results. Because the model is based on accurate terrain and building height data, more detailed assessments of environmental conditions or energy-related phenomena can be carried out. At the same time, this complete process also provides a practical reference for digital twin-oriented urban modeling, especially suitable for application scenarios involving large-scale visualization or parametric simulation in virtual environment[26].

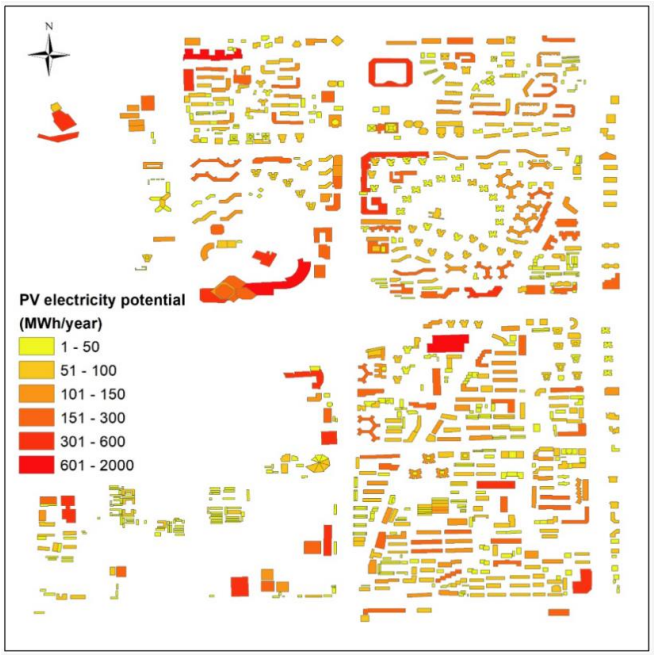


Figure 16a. Annual PV potential map in Chaoyang District, Beijing

Source: Lichao Wang et al., *High resolution photovoltaic power generation potential assessments of rooftop in China.* 2022

Figure 16b. Different types of rooftops (flat rooftops are shown in red and sunny hips and shade hips of pitched roofs are shown in yellow and green, respectively)

Source: Lichao Wang et al., *High resolution photovoltaic power generation potential assessments of rooftop in China.* 2022

3. Stakeholder Collaboration:

The 3D models and analytical results produced in ArcGIS Pro can be published through web-based platforms such as ArcGIS Online, allowing planners, developers, and community members to review and comment on ongoing projects. This type of dissemination supports coordinated decision-making by ensuring that all stakeholders have access to the same spatial information and can contribute feedback during the planning process.

A practical example comes from the City of Mesa, where the Planning Department has adopted the ArcGIS ecosystem—specifically ArcGIS Urban, CityEngine, and Experience Builder—to strengthen public engagement and internal collaboration. Through tools such as the "Active Development Sites"

interactive web map (Figure 17), users can explore up-to-date project data and 3D visualizations. The platform enables citizens, professionals and municipal workers to realize the progress of tracing projects in real time, thus enhancing transparency and promoting more informative public participation.

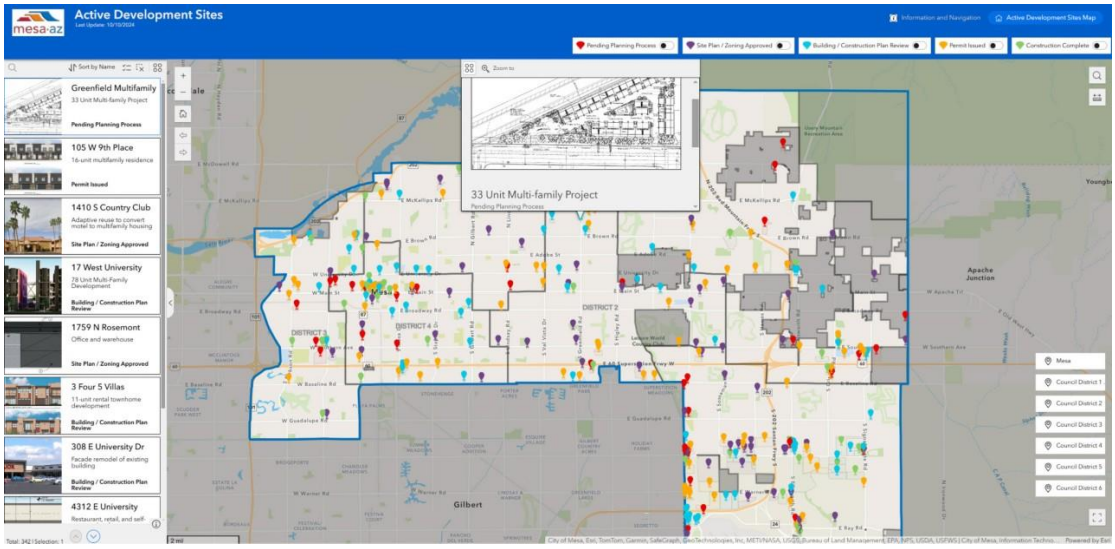


Figure 17. The city of Mesa's " Active Development Sites" interactive map shows ongoing urban construction projects across the city.

Source: *https://experience.arcgis.com/experience/e7c4f8f4201641919a28b28452fa095c*

During the 2050 General Plan update, an immersive experience combining GIS and Unreal Engine enabled over 350 students to explore future city transformations in mixed reality (Figure 18).

Figure 18. The immersive experience interface presents the planned street renovation scheme in a way close to the real scale, so that users can intuitively understand the future urban changes.

Source:*https://www.esri.com/en-us/lg/industry/government/stories/city-of-mesa-revolutionizes-urban-planning-projects-with-gis*

This case illustrates how integrating GIS-based modeling and visualization tools fosters a transparent, interactive, and participatory framework for collaborative urban planning.

- 3.2 City Engine (ESRI)

- 3.2.1 Software Overview

CityEngine [27] is a specialized software application developed by Esri, designed for the parametric modeling of urban environments. Compared to ArcGIS Urban, CityEngine focuses on rule-based, large-scale 3D city modeling through CGA procedural generation, whereas ArcGIS Urban is a web-based platform designed for planning scenarios, zoning evaluation, and interactive urban decision-making rather than detailed procedural modeling. CityEngine has become a commonly adopted tool in urban planning, architectural design, and digital media production because it can generate extensive and detailed 3D city models with a high level of automation (Figure 19). Its workflow is built around a rule-driven modeling system in which users specify parameters such as building height, roof configuration, or street form, and the software interprets these instructions to construct large urban environments.

The modeling process is controlled by CGA, a scripting language developed specifically for City Engine to define how building geometry is generated and combined. Through the combination of parameter control and programmatic generation, CityEngine can quickly build data-based urban scenes for simulation, visualization and spatial analysis.

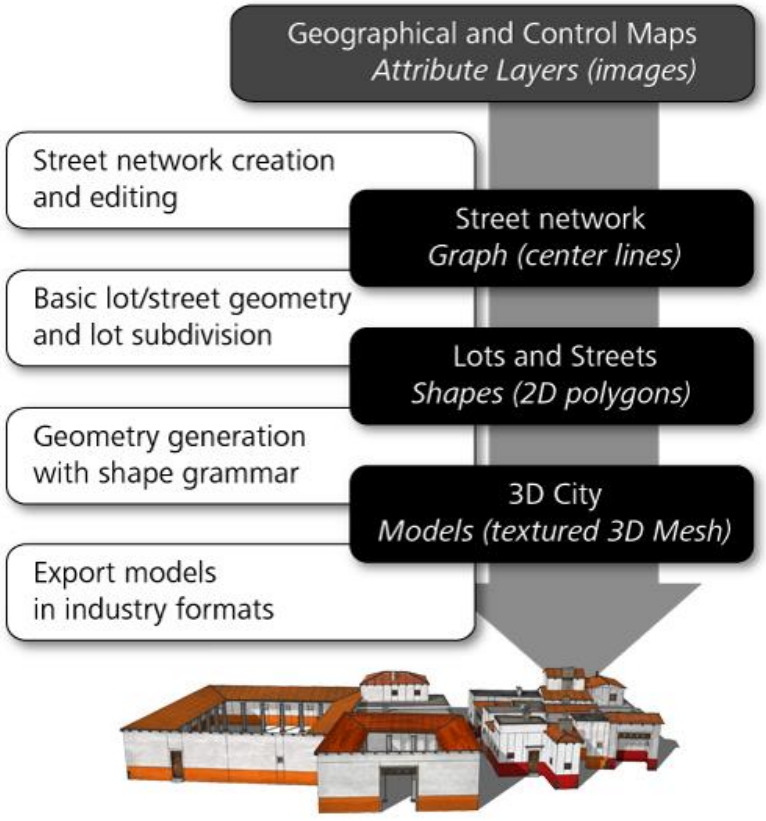


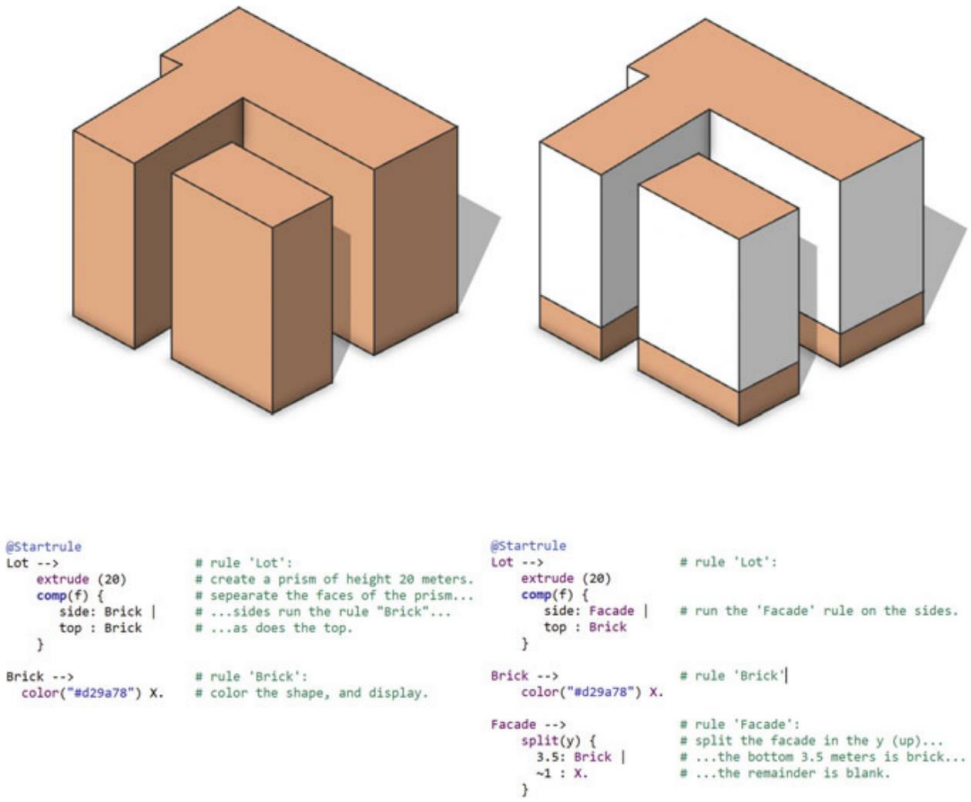
Figure 19. Overview of the CityEngine modeling pipeline. Black boxes illustrate data types (layers) and white boxes the operations to create them. Typically, in the first step, the street network is created, afterwards the resulting blocks are subdivided into lots. Finally, the 3D models of the buildings are generated using the CGA rules. The output of CityEngine is polygonal building models. (Esri)

- 3.2.2 Advantages of CityEngine

1. Parametric Generation:

One of the core strengths of CityEngine is its parametric generation capability. This feature allows users to generate complex urban environments efficiently by applying a set of customizable rules, drastically reducing the time required for manual modeling. As shown in Figure 20, the two buildings with distinct appearances are each controlled by different CGA rule files. The rule file on the left defines the extrusion of a lot along the Y-axis by 20 meters, forming a three-dimensional mass. Each face of this volume is decomposed into different types and assigned a red brick material. In contrast, the rule on the right introduces façade stratification: the lower 3.5 meters are designated as a brick base, while the upper portion remains a blank wall surface[28].

This parametric approach is particularly advantageous when constructing large-scale urban landscapes, because traditional modeling methods often require a lot of time and labor (Esri, 2023).



```
@Startrule
Lot -->                 # rule 'Lot':
    extrude (20)        # create a prism of height 20 meters.
    comp(f) {           # sepearate the faces of the prism...
       side: Brick |    # ...sides run the rule "Brick"...
       top : Brick      # ...as does the top.
    }

Brick -->               # rule 'Brick':
  color("#d29a78") X.   # color the shape, and display.
```

```
@Startrule
Lot -->                 # rule 'Lot':
    extrude (20)
    comp(f) {
       side: Facade |   # run the 'Facade' rule on the sides.
       top : Brick
    }

Brick -->               # rule 'Brick'
    color("#d29a78") X.

Facade -->              # rule 'Facade':
    split(y) {          # split the facade in the y (up)...
      3.5: Brick |      # ...the bottom 3.5 meters is brick...
      ~1 : X.           # ...the remainder is blank.
    }
```

Figure 20. CGA rule files control different building facades

Source: Kelly, Tom. *CityEngine: An Introduction to Rule-Based Modeling*. 2021.

2. Flexibility and Customization:

CityEngine offers extensive flexibility through its rule-based system. Users can customize various parameters, such as building height, roof types, and street layouts, to generate models that meet specific project requirements. This adaptability is critical for urban planners and architects who need to explore multiple design scenarios quickly [29] (Müller et al., 2006). For instance, within the Inspector panel of CityEngine, users can adjust parameters such as Level of Detail (LoD), height

and roof type according to specific design requirements(Figure 21).

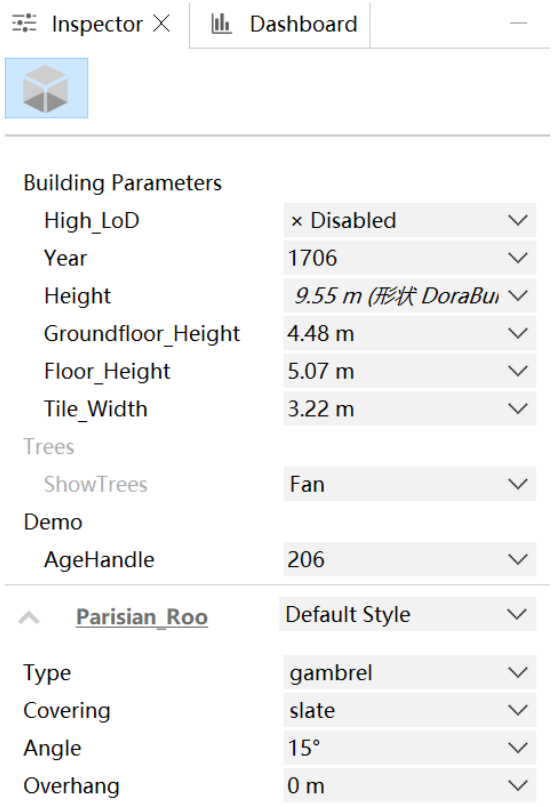


Figure 21. Building parameters set in CityEngine

3. Tight Integration with ArcGIS:

CityEngine can utilize GIS vector data (such as shape) as the fundamental data for modeling. Data like land parcels, building boundaries, and road centerlines can all be directly loaded into CityEngine for use. Whether it is measurement data, analysis results, or planning schemes, they can all be easily imported into the CityEngine scene, ensuring the accuracy of the model data. Additionally, CityEngine conducts terrain - based modeling, making the created models seamlessly fit the terrain. This integration process ensures that the model is visually realistic. It also ensures that the model is geographically accurate. Which makes CityEngine an ideal tool for producing highly detailed, realistic projects.

4. Visualization and Analysis:

CityEngine provides advanced visualization tools that facilitate the analysis and presentation of urban models. Features like real-time rendering and the ability to export models to various platforms allow users to effectively communicate design concepts and urban plans to stakeholders. As illustrated in Figure 22, in the previously mentioned case study conducted by the Planning Department on the urban design of the City of Mesa, in its Transit-Oriented Development (TOD) initiative, CityEngine tools were used to simulate urban design scenarios and assess impacts on land use and mobility, allowing residents and agencies to engage in data-informed decision-making. This capability enhances the overall decision-making process in urban planning (Esri, 2023).

Figure 22. Using ArcGIS Urban in conjunction with ArcGIS CityEngine, a three-dimensional model of this public transit-oriented development plan was generated. These three-dimensional contents enable the planning and design scheme to be evaluated in spatial dimension.
Source:*https://www.esri.com/en-us/lg/industry/government/stories/city-of-mesa-revolutionizes-urban-planning-projects-with-gis*

CityEngine stands out as a powerful tool in the realm of urban modeling and simulation, offering unique capabilities that streamline the design and planning process, while also enhancing the precision and quality of the resulting models.

- 3.3 SketchUp

- 3.3.1 Software Overview

Because some landmark buildings and irregular buildings in the study area are complex in shape and difficult to be reliably generated by CGA rule program, their geometric shapes cannot be accurately restored by rule modeling, so manual modeling is carried out in SketchUp instead[30]. This method provides better control over complex geometries and detailed height details. These models are then imported into the ESRI environment to maintain consistency with the overall geographic information modeling process.
SketchUp as the complex building modeling software, which is developed initially by Last Software and now owned by Trimble Inc., is a user-friendly 3D modeling software designed for a wide range of applications, from architectural design and interior planning to landscape architecture, product design, and even 3D printing. Unlike complex professional 3D modeling tools that require extensive technical expertise, SketchUp emphasizes simplicity and intuitiveness, enabling users to transform ideas into 3D geometries quickly through its "push-pull" core functionality—a unique feature that allows for the easy extrusion of 2D shapes into 3D forms. It is defined not only as a modeling tool but also as a collaborative platform that bridges the gap between conceptual design and practical implementation, catering to both beginners and industry professionals.

- 3.3.2 Characteristics of SketchUp

As a user-friendly 3D modeling software, SketchUp's functional advantages lie in its simplicity, efficiency, and extensibility—all designed to bridge the gap between conceptual design and practical implementation, while catering to both beginners and professionals. Below is an integrated overview of its key functional features.

The core advantage of SketchUp is that its tools can greatly simplify the 3D geometric modeling process while ensuring flexibility, so that users can easily transform design ideas into visual models.

1. Push-Pull Tool (Signature Feature):

The most representative function of SketchUp is that it can directly manipulate two-dimensional shapes (such as rectangles, circles, polygons, etc.). By pushing (stretching) or pulling (compressing) a planar shape, you can immediately convert it to a 3D mass. For example, users can turn a two-dimensional square into a cube, a rectangular plane into a wall, or a circular outline into a cylinder. This tool eliminates the need for complex parameter settings, making design iterations fast and efficient.

2. Real-Time Visualization Aids:

Built-in tools like Shadows and Fog let users simulate real-world lighting conditions (e.g., sunlight angles at different times of day) and atmospheric effects. This helps preview how a design will look in actual environments during the modeling process. For higher-quality outputs, the software also integrates seamlessly with third-party rendering tools (e.g., V-Ray, Enscape) to generate photorealistic images, animations, and even virtual walkthroughs—expanding its utility for client presentations or project demonstrations.

3. 3D Warehouse:

An integrated online library of free and paid 3D models (e.g., furniture, trees, building components, appliances) created by the global SketchUp community. Instead of recreating common objects from scratch, users can search, download, and directly import these models into their projects—saving hours of modeling time. The library also supports uploading custom models, enabling teams to share project-specific components, some of the architectural models in this thesis project were obtained from 3D Warehouse (Figure 23).

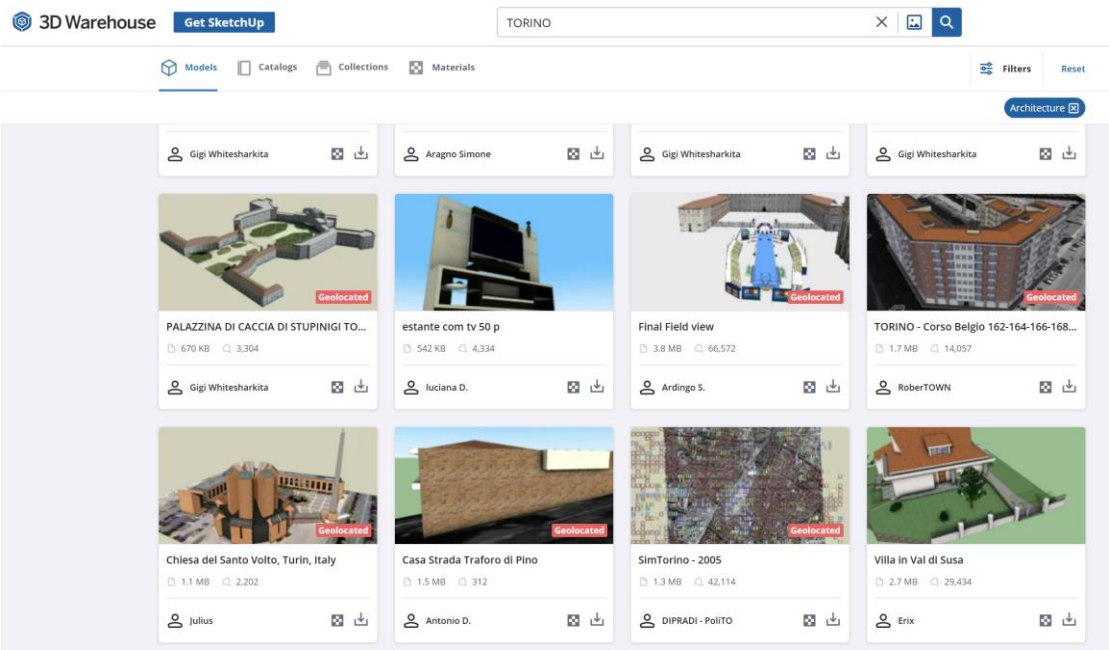


Figure 23. 3D Warehouse online library

Source:*https://3dwarehouse.sketchup.com/search/models?q=TORINO&domain=Architecture&order=relevance&direction=desc*

4. Broad File Format Support:

It natively supports popular design formats, including:SKP (SketchUp's proprietary format, for saving and sharing models with other SketchUp users); DWG/DXF (for seamless integration with AutoCAD, a staple in architecture and engineering); OBJ (for importing/exporting models to 3D rendering, animation, or game design software); STL (for 3D printing, enabling users to export models directly to 3D printers or slicing software).

- 3.4 Cesium for Unreal Engine 5

- 3.4.1 Software Overview

Cesium for Unreal Engine 5 (Cesium for UE5)[31] is a sophisticated, open-source integration plugin that serves as a critical technical intermediary between Cesium's enterprise-grade geospatial data processing ecosystem and UE5—a leading real-time 3D (RT3D) development platform renowned for its advanced graphics rendering, physics simulation, and interactive content creation capabilities in both industrial and entertainment domains. Conceptually framed as a "geospatial-RT3D convergence tool," this plugin addresses a longstanding industry challenge: the disconnection between high-precision geospatial datasets—such as digital elevation model (DEM), orthophotos, and LiDAR-derived point clouds—and RT3D environments that enable immersive, real-time exploration. Unlike standalone geospatial visualization tools (e.g., ArcGIS Pro, QGIS) that prioritize data accuracy but lack robust RT3D rendering or dynamic simulation, or generic RT3D engines that

excel at interactive content but require manual, error-prone geospatial data integration, Cesium for UE5 resolves this dichotomy by embedding Cesium's geospatial data management and coordinate system expertise directly into UE5's development workflow.

The core capability of Cesium for UE5 is to enable users to seamlessly import, visualize and interactively manipulate geospatial data sets in the real-time environment of UE5-including global terrain models, high-resolution satellite / aerial images, three-dimensional building outlines, vector geographic data (roads, administrative boundaries, etc.), and custom geospatial assets. This integration is not only a simple technical " bridge", but also an enhancement layer with transformational significance, which makes UE5 transform from a general real-time 3D engine to a professional platform specially oriented to geospatial.
In this workflow, the geometrical accuracy and spatial positioning accuracy of geospatial data are completely preserved, which is very important for industrial applications such as digital twins. It also takes advantage of UE5's advanced features: Nanite Virtualization Geometry, which renders massive amounts of geospatial data without sacrificing performance,Lumen Global Lighting System (for realistic lighting of real-world scenes), Chaos physical systems (for simulating real-world phenomena such as weather, traffic, or structural dynamics on geospatial surfaces).

In a related study, Toni Rantanen et al. built a game engine environment based on the 3D city model of Helsinki. The research team uses WGS84 coordinate system to establish an operable real-world coordinate frame in virtual space (Figure 24), and ensures that the 3D city model can be completely aligned with typical physical engines and lighting systems, thus realizing accurate spatial registration and realistic visualization effects. Thanks to the high-quality visualization capability and real-time rendering performance of UE5, this platform has obvious potential for application in urban scale digital twin simulation[32].

Figure 24. Helsinki 3D city model view in Unreal Engine
Source: Toni Rantanen et al., *Open Geospatial Data Integration in Game Engine for Urban Digital Twin Applications.* 2023

Notably, Cesium for UE5's open-source architecture distinguishes it from proprietary geospatial-RT3D integrations, as it allows for customization and extension to meet domain-specific requirements—for example, integrating with third-party sensor data platforms (e.g., IoT devices for real-time infrastructure monitoring) or adapting to specialized coordinate systems (e.g., local projected coordinate systems for construction sites). This flexibility, combined with its adherence to industry geospatial standards (e.g., OGC-compliant data support, global coordinate system such as UTM WGS84 compatibility), positions Cesium for UE5 as a foundational tool in the emerging field of "geospatial metaverses"—environments where real-world geographic context is merged with digital interactivity to drive decision-making, training, and engagement.

- 3.4.2 Characteristics of Cesium for UE5

1. Integration and Streaming of High-Resolution Geospatial Data
Cesium for Unreal lets developers seamlessly integrate high-resolution geospatial datasets—3D Tiles, terrain elevation models, photogrammetric city meshes included—and stream, visualize vast real-world environments in real time. This capability proves especially important for urban digital twins, where accuracy and spatial fidelity are the backbone of both analytical work and visualization efforts.

2. 3D Model Data Integration
Datasmith, developed by Epic Games, is an important conversion tool for importing models from CityEngine and other design software into Cesium for UE5. It preserves the geometry, materials, and hierarchy intact during file conversion. This ensures that the parametric model still maintains accuracy and visual quality in a real-time environment. When Datasmith is used in conjunction with Cesium's georeferenced environment, the model accurately aligns global coordinates and coexists well with data such as topography, 3D Tiles, and satellite imagery. Which capability enables efficient, interoperable digital twin workflows between GIS-based modeling and real-time visualization.

3. Advanced Visualization and Immersive Rendering
Take the Helsinki 3D City initiative discussed earlier. By drawing on Unreal Engine's Nanite virtualized geometry and Lumen global illumination, Cesium for Unreal supports advanced lighting, shadowing, and atmospheric effects—crafting photorealistic cityscapes with cinematic quality. These features boost the immersive realism of urban digital twins, closing the divide between geospatial accuracy and visual storytelling.

In summary, Cesium for Unreal Engine 5 stands out for urban 3D modeling because it merges real-world geospatial data and Unreal Engine's powerful visualization, interaction capabilities. This combination lets users build accurate, detailed, and immersive urban environments that work for a broad array of uses—from city planning to virtual reality simulations.

- 3.5 Data interoperability between platform

In the workflow of digital urban modeling, multiple software platforms are often involved, each with its own specialization in data acquisition, processing, modeling, and visualization. ArcGIS Pro is primarily used for spatial data management and preprocessing, while CityEngine provides parametric modeling capabilities based on rule-driven approaches. SketchUp is often utilized for creating detailed landmark buildings that require manual fine-tuning, and Cesium for UE5 serves as the final rendering and visualization environment for interactive presentation.

A central challenge in this workflow is data interoperability. Each software platform supports its own independent input and output formats, so to achieve smooth data exchange, it is necessary to ensure that geometric accuracy, semantic consistency and visual fidelity are not destroyed during the conversion process. Table 3 below lists the formats compatible with ArcGIS Pro, CityEngine, SketchUp and Cesium for UE5, and summarizes how geospatial data and 3D data are shared between these platforms to show how they work together to build a coherent city model.

Software	ArcGIS pro	SketchUp	CityEngine	Cesium for UE5
input data formats	-Shapefile (.sh*) -File Geodatabase (.gdb) -Raster/DEM(.tif, .img) -GeoJSON (.geojson) - CAD data(.dwg, .dxf)	- SKP (.skp) - Collada (.dae) - 3DS (.3ds) - DWG/DXF (.dwg, .dxf)	- Shapefile (.shp) - Geodatabase (.gdb) - DEM (.tif) - OBJ (.obj) - Collada (.dae) - FBX (.fbx)	- 3D Tiles (.json+.b3dm/ .i3dm/ .pnts) - glTF/GLB (.gltf, .glb) - FBX (.fbx) via Unreal import -Datasmith
processing / export formats	- Shapefile (.sh*) - Geodatabase (.gdb) - Raster (.tif) - Scene Layer Package (.slpk) - 3D Object (.obj)	- Collada (.dae) - OBJ (.obj) - FBX (.fbx)	- OBJ (.obj) - FBX (.fbx) - Collada (.dae) - Esri Scene Layer Package (.slpk) - 3WS (.3ws) for web scene - glTF/GLB (.gltf, .glb) -Datasmith	Native rendering in UE5 Supports Cesium 3D Tiles Lumen global illumination Chaos physics Streaming

Table 3 Data format Interoperability between Software

ArcGIS Pro is mainly used for geospatial preprocessing and data management in urban modeling projects. Its core functions cover cleaning raw geospatial datasets, managing coordinate systems and projections, generating and refining elevation models, and preparing vector layers like parcel boundaries or road networks. These preprocessing steps let ArcGIS Pro guarantee the spatial data is accurate, standardized, and compatible for subsequent import into CityEngine—laying a reliable foundation for parametric urban modeling.

CityEngine can import GIS data from ArcGIS Pro, such as parcel polygon and DEM grid elevation data. It automatically generates a city model with both geometric and semantic information through parametric modeling rules (such as CGA rules). The software gives users the ability to iterate quickly and explore different designs with flexibility, while preserving imported geographic information

intact. In addition, CityEngine supports exporting models to a variety of formats. For example, glTF can be used to visualize Cesium; FBX can be integrated with UE5 to make the subsequent advanced rendering process smoother.

SketchUp is typically used to construct elaborate landmarks, such as churches, public buildings, or structures of cultural interest. This type of building requires manual adjustment and higher accuracy, so SketchUp modeling is more appropriate.Models can be exported to Collada (. dae) or FBX (. fbx) format and imported into CityEngine for alignment with parametrically generated city textures. In this way, the geometric details and material textures of landmark buildings can be preserved at the same time, so that they can naturally blend with the model generated by CityEngine in the same spatial environment.

CityEngine also plays a key role in connecting function between fine 3D modeling and large-scale visualization platform. It allows users to convert integrated models - including urban areas generated based on parametric rules, and landmarks imported from SketchUp - into glTF or 3D Tiles format for Cesium for UE5 streaming loading. For complex assets (especially models from SketchUp), the system will transform and adjust them according to parameterization rules to ensure accurate spatial location, correct scale and realistic appearance. Such an interoperability framework significantly improves overall efficiency, reduces a lot of repetitive manual work, and enables urban planners and researchers to simulate and visualize large-scale urban environments with high precision and high visual quality.

Data interoperability between ArcGIS Pro, CityEngine, SketchUp, and Cesium for UE5 relies on choosing the right exchange formats. ArcGIS Pro supplies the geospatial base data in .shp and DEM.tif formats, and CityEngine uses these for parametric model generation. SketchUp provides detailed landmark models exported as .obj files, which can be integrated into CityEngine for urban context alignment. The final stage leverages Cesium's support for Datasmith models and 3D Tiles, enabling efficient streaming and rendering within UE5. (Figure 25).

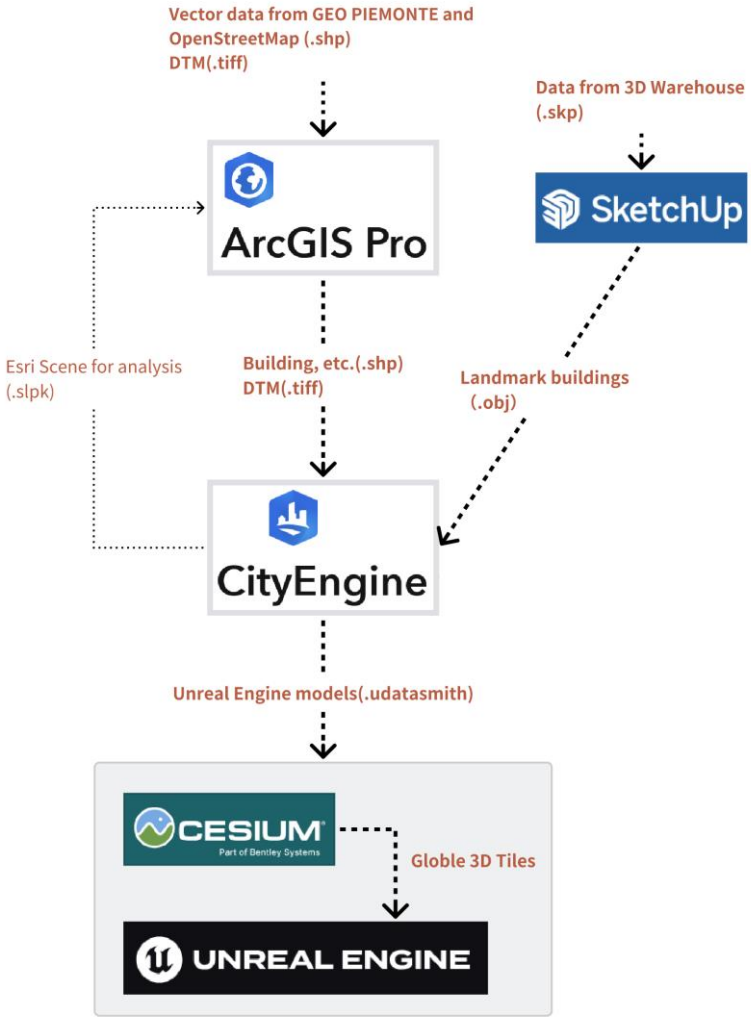


Figure 25. The data flow among the different software platforms involved in this thesis

This interoperability framework not only ensures the smooth flow of spatial and 3D model data across platforms but also significantly enhances the scalability and efficiency of urban digital twin projects. By establishing a standardized data exchange strategy, the workflow minimizes format incompatibilities and maximizes the fidelity of urban visualization outcomes.

4. Methodology for constructing three-dimensional city model

- 4.1 Data access for 3D modeling

- 4.1.1 DEM Data

Definition:

Urban 3D landscape modeling is based on the acquisition of image data, vector data, DEM data, attribute data, and texture data. It builds a three-dimensional model of the main elements of the city by comprehensively using computer graphics technology, 3D GIS technology, and 3D modeling technology. How to obtain high-quality basic data at a low cost is a key issue in urban 3D landscape modeling.

DEM is a two-dimensional data that stores elevation information, generally represented by an elevation matrix. Digital elevation model data plays a very important role in urban three-dimensional landscape modeling and is the main data source for building urban terrain models. At present, there are many main ways to obtain DEM data, which can be divided into the following types according to the collection method.

1. Airborne and Terrestrial LiDAR Scanning:

LiDAR (Light Detection and Ranging) technology - both airborne and ground-based systems - is one of the most accurate methods available for obtaining high-resolution elevation data. Airborne Laser Radar (ALS) and LiDAR mounted on UAV can cover medium and large areas efficiently; Ground-based laser scanning (TLS) can capture fine micro-topography and architectural details with millimeter accuracy (Figure 26). These high-density point cloud data are especially suitable for digital twin construction on urban scale, because they can accurately present building facade, vegetation structure and subtle surface fluctuation, which is a key function for the geometric authenticity of the model[33].

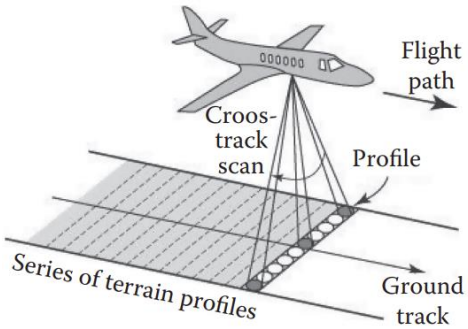


Figure 26: The coverage of a terrain strip from an airborne platform is accomplished by acquiring a sequence of elevation profiles measured transversely to the flight path.

Source: Jie Shan et al., *Topographic Laser Ranging and Scanning Principles and Processing*. 2018

2. Satellite-Based Radar and Optical Photogrammetry:

Global DEM datasets, such as SRTM, ASTER GDEM, and Copernicus DEM, are derived from radar interferometry (InSAR) or satellite optical stereo imagery. These techniques allow the generation of

continuous elevation surfaces at continental to global scales, supporting large-scale terrain modeling and hydrological analysis [34]. In particular, satellite optical stereo images—acquired from platforms such as SPOT, Pleiades, or WorldView—enable high-resolution photogrammetric reconstruction of the Earth's surface through multi-angle image correlation (Figure 27).

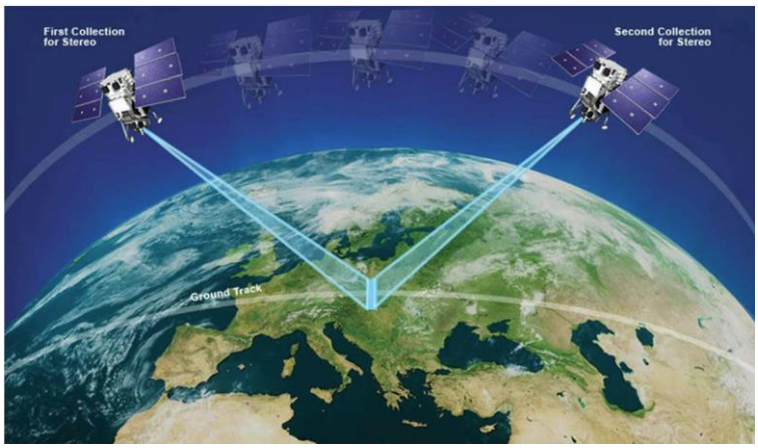


Figure 27. In-track stereo collection mode. The sensor collects two shots (45-90 seconds apart) of the same location by adjusting its camera.
Source:*https://www.pgc.umn.edu/guides/stereo-derived-elevation-models/introduction-to-stereoscopic-imagery/*

Compared with radar-based InSAR methods, optical stereo DEMs provide finer spatial detail and superior visual interpretability, making them valuable for urban-scale digital twins and landform refinement within GIS-based workflows. Although their vertical precision may be affected by illumination and atmospheric conditions, their broad availability and high geometric fidelity make them indispensable complements to radar-derived datasets in multi-scale geospatial modeling.

3. Digital Photogrammetry and Structure-from-Motion (SfM):
High-resolution DEMs can also be generated through aerial or UAV-based photogrammetry (Figure 28). Structure-from-Motion (SfM) algorithms reconstruct 3D geometry by matching overlapping images, providing centimeter-level accuracy at relatively low cost. This approach is highly efficient for localized areas such as parks, construction sites, or cultural heritage zones, making it a preferred method in urban-scale architectural research[35]

Figure 28. UAVs ground sampling
Source:*https://garudsurvey.com/revolutionizing-land-mapping-with-drone-survey-in-chennai-garud-survey-private-limited/*

4. Conventional Ground Surveying and Map Digitization:

Traditional surveying techniques—GNSS measurements, total stations, and contour digitization included—still prove useful for small-scale, high-precision studies or verifying remotely sensed data. These methods often require a lot of manual operation, so they are not suitable for large areas or places that are difficult to reach.

Another method is RTK (Real Time Dynamic Positioning) measurement. This is a measurement technology that can achieve centimeter-level accuracy, which is much more accurate than traditional GPS. RTK works with the Global Navigation Satellite System (GNSS) and can use GPS, GLONASS, Galileo, Beidou and other satellite signals. It combines the principle of satellite positioning with real-time differential correction data to achieve high-precision measurement. RTK measurement relies on two core devices: RTK base station and RTK mobile station (Figure 29).

The base station is a static point with pre-surveyed, known coordinates. It continuously tracks GNSS signals and calculates correction data—accounting for errors from ionospheric and tropospheric interference, satellite clock inaccuracies, and orbital deviations. This correction data gets sent in real time to the RTK rover.

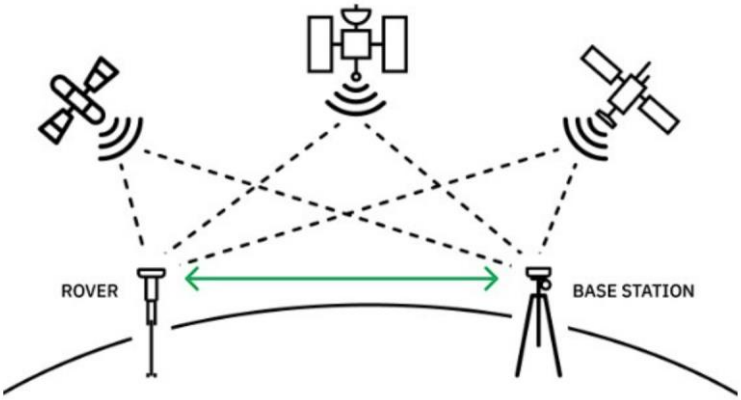


Figure 29. Two key components: RTK base station and RTK rover
Source: *https://www.gpsworld.com/understanding-gnss-correction-methods/*

Data sources:

In this study, selecting the DTM data of Turin and searched for the corresponding location data on the Geoportale Piemonte (Figure 30), as the roads and buildings located above the ground surface are processed independently as vector layers in this research, only the bare-earth terrain is utilized for subsequent modeling. The DTM covers the entire regional territory and was acquired with a uniform methodology (LIDAR) in level 4 standard. The grid resolution (step) is 5 m, with an altitude precision of ±0.30 m (±0.60 m in the areas of lower precision, corresponding to wooded and densely urbanized areas) (Figure 31).

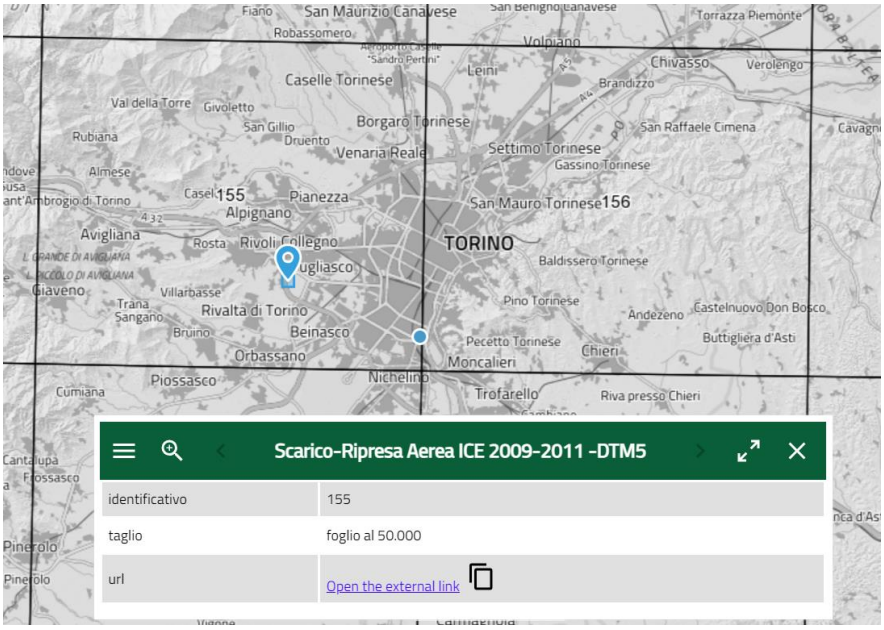


Figure 30. ICE AERIAL SHOT 2009-2011 - DTM 5

Source:*https://www.geoportale.piemonte.it/geonetwork/srv/ita/catalog.search#/metadata/r_piemon:224de2ac-023e-441c-9ae0-ea493b217a8e*

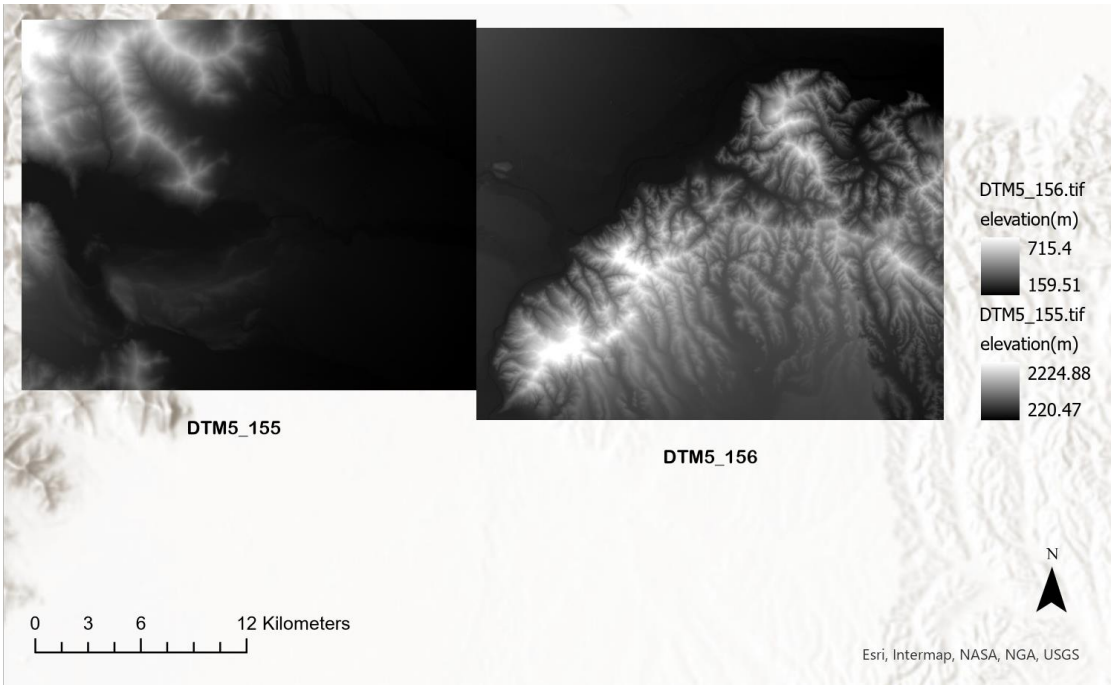


Figure 31. Importing DTM images into ArcGIS Pro

- 4.1.2 Orthophoto

Definition:

An orthophoto is a digital aerial or satellite image that has been systematically corrected so that objects and features across the entire scene appear in their correct planimetric positions;

Orthophotos align perfectly with a planimetric map of an area—this accuracy comes from the orthorectification process. Without orthorectification, images have geometric distortions—users can't measure distances, angles, positions, or areas directly and accurately.

The elevation difference of the surface topography and the tilt of satellite or aerial sensors can cause the position of the features in the image to shift. The more complex the terrain, the more obvious the geometric distortion. Therefore, when acquiring new high-resolution images in rugged areas, it is often necessary to use sensors with a larger viewing angle to reduce the displacement caused by perspective.

To further eliminate residual distortion, the orthographic correction process uses a DEM to provide the necessary terrain elevation information for geometric correction (Figure 32). In addition, ground control points (GCP) can further improve the geometric accuracy of images-they are used to calibrate the correction process and verify the accuracy of the final orthophoto.

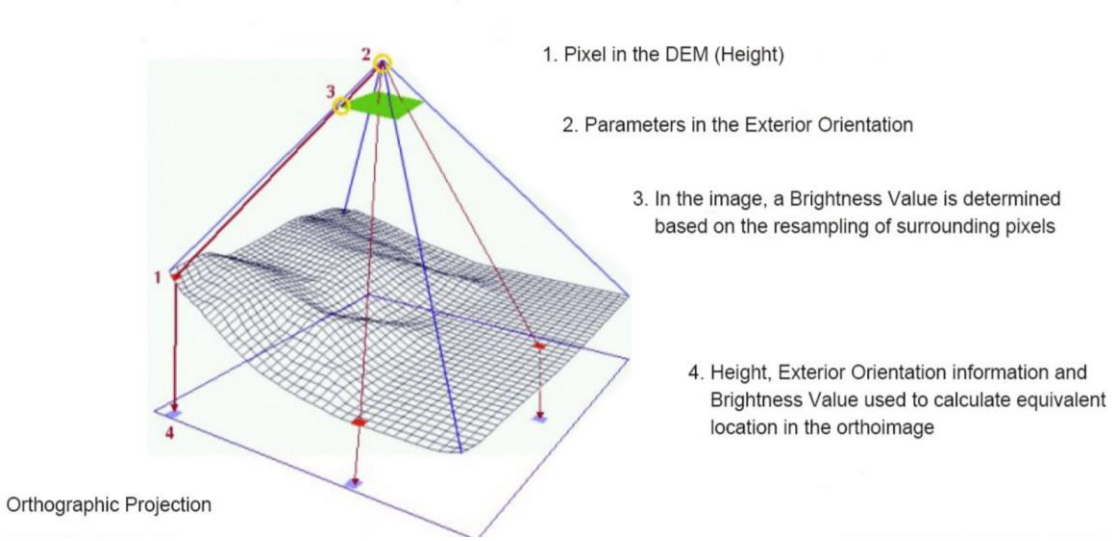


Figure 32. Orthorectification process of remote sensed Image data
Source: *https://www.satimagingcorp.com/services/orthorectification/*

Data access:

On the Geoportale Piemonte, it is possible to search for the dataset Ortofoto digitale colore 2023 (Figure 33). This platform provides access to a WMS basemap of the city of Turin.

Figure 33. Torino Digital color orthophoto 2023 (GeoTIFF file)
Source:*https://www.geoportale.piemonte.it/geonetwork/srv/ita/catalog.search#/metadata/c_l219:e9b9f64d-ae84-4ed8-82d5-32113a3c0c0d*

Since access to online basemaps is sometimes restricted and coordinate systems cannot be preserved when printing from ArcGIS Online, an alternative approach is to download orthophotos (WMS server data) from ArcGIS Online directly within ArcGIS Pro. To export the downloaded imagery as a raster file for subsequent use in CityEngine, the GeoTIFF Map Export tool in ArcGIS Pro can be employed. This allows the generation of high-resolution (almost 0.5 meters) true color image suitable for integration into 3D modeling and visualization workflows (Figure 34) .

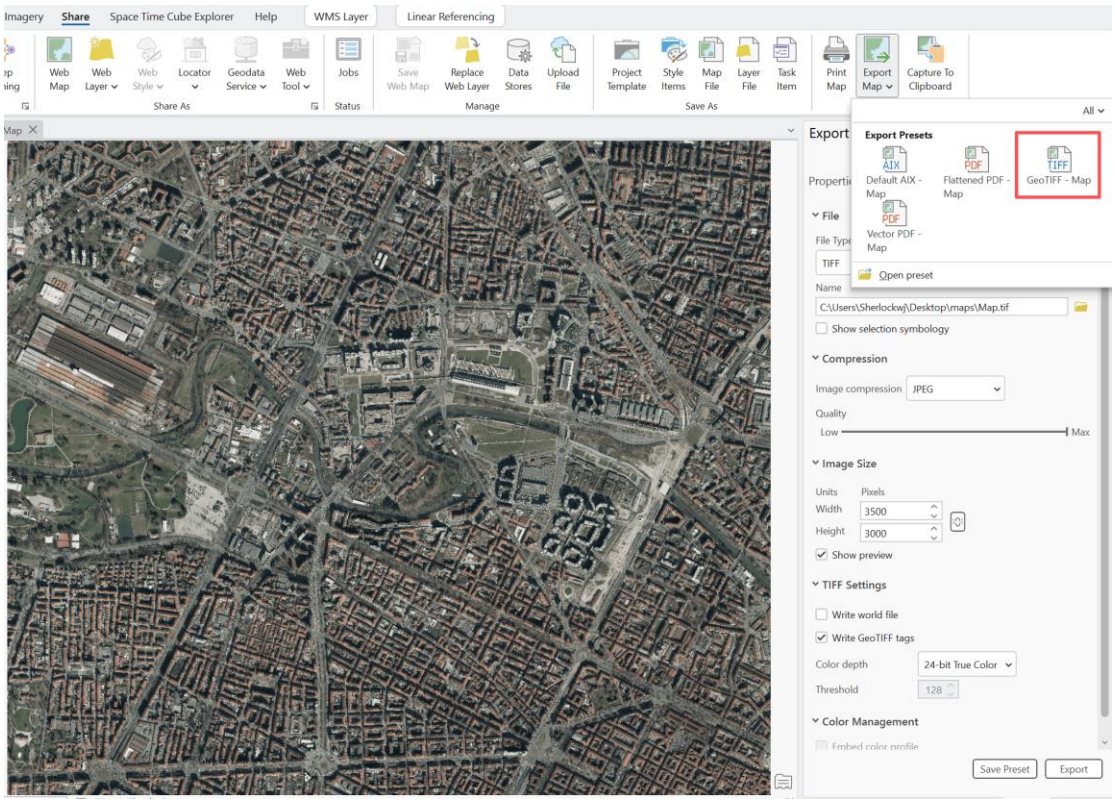


Figure 34. Convert WMS Server data to Raster layer in ArcGIS Pro

- 4.1.3 Vector Data

Definition:

A shapefile is an Esri vector data storage format for storing the location, shape, and attributes of geographic features. It is stored as a set of related files and contains one feature class. Shapefiles often contain large features with a lot of associated data and historically have been used in GIS desktop applications, geographic features in a shapefile can be represented by points, lines, or polygons (areas).

The primary way to make shapefile data available for others to view through a web browser is to add it to a .zip file, upload it, and publish a hosted feature layer. The .zip file must contain at least the .shp, .shx, .dbf, and .prj files components of the shapefile. Below is an example of how shapefiles appear in Catalog pane Figure 35.

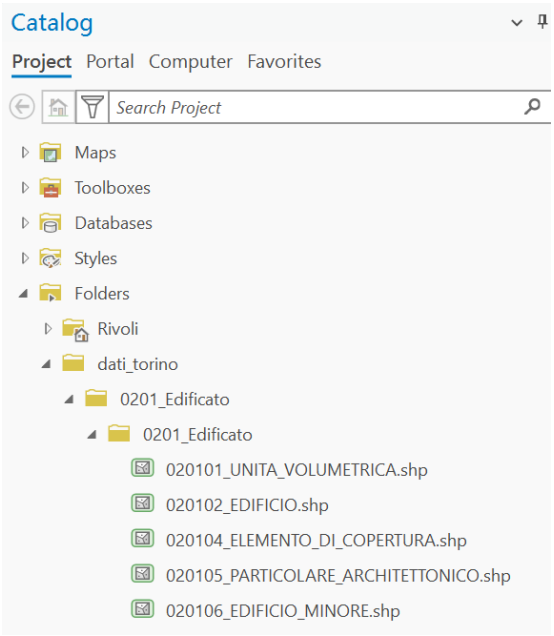


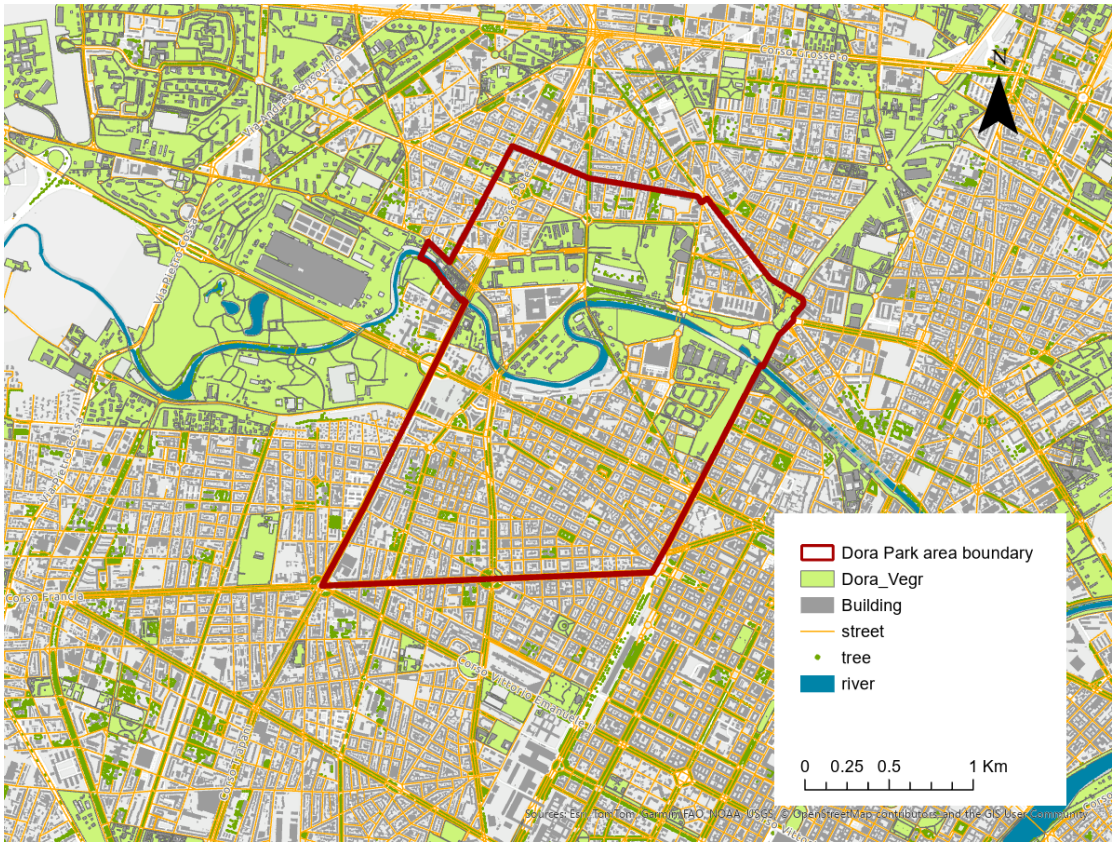
Figure 35. Torino buildings shapefiles in Catalog pane

Data sources:

Most of the shp data used in this thesis comes from OpenStreetMap (OSM)—a collaborative open-source mapping project launched in 2004. It lets users around the world contribute to, edit, and freely use geospatial data [36]. Unlike commercial mapping platforms, OSM offers free access to geographic information like roads, buildings, administrative boundaries, and land use data.

As OSM adopts a community-driven update model, its data set is continuously supplemented and improved - especially in urban areas, where local contributors will continue to add detail and improve data accuracy. Thanks to its Open Database License (ODbL), OSM has become one of the most widely used geospatial data sources for academic research, urban studies and digital twin applications.

To get the needed data, first select the target region on the GeoFabrik platform, which distributes OpenStreetMap datasets: start with Europe and gradually narrow it down to northwest Italy, then download the free shapefile. The downloaded ZIP file finally includes vector files for elements like trees, roads, water systems, railways and Dora Park boundary area. Because the building in OSM has only outline and no height attribute, which is not available for parametric modeling later, the building vector file with height data will be obtained from the municipality of Torino (Figure 36).



Geographic Coordinate System	WGS 1984
Authority	EPSG
Year	2024

Figure 36. Shapfile from GIS OSM

- 4.1.4 3D model with SketchUp

Definition:

A 3D model created with SketchUp represents the geometric and spatial configuration of architectural or environmental objects using surface-based modeling. The software allows intuitive creation, editing, and visualization of three-dimensional forms, supporting georeferencing and data exchange with other platforms such as CityEngine and Unreal Engine for simulation and visualization purposes.

Data sources:

Because there are many landmark buildings such as churches in the selected area, in order to enhance the recognition of the site and the final building accuracy, this part of the model in the thesis was obtained from 3D Warehouse, which is an integrated online library of free and paid 3D models (e.g., furniture, trees, building components, appliances) created by the global SketchUp community. Instead of recreating common objects from scratch, users can search, download, and directly import these models into their projects—saving hours of modeling time. The library also

supports uploading custom models, enabling teams to share project-specific components, some of the architectural models in this thesis project were obtained from 3D Warehouse
Models obtained from online 3D model libraries are user-generated and primarily intended for visualization and approximate measurement. Most of these models lack georeferencing and standardized metadata; therefore, when integrating them into real-world contexts, it is necessary to verify whether the building dimensions and proportions comply with actual standards. While high-precision 3D metric surveys can indeed be produced through various professional techniques, such methods exceed the objectives of this research. The intention here is not to achieve a fully accurate digital representation of the built environment, but rather to demonstrate how externally sourced models can be integrated into a GIS-based workflow to support the methodological development of a broader digital-twin process.

From the 3D Warehouse, it can be found that Curia arcivescovile di Torino and Environment Park are included in the scope required by the thesis Figure 37. These buildings have Geolocated features, which can align coordinates and then download models as skp files.

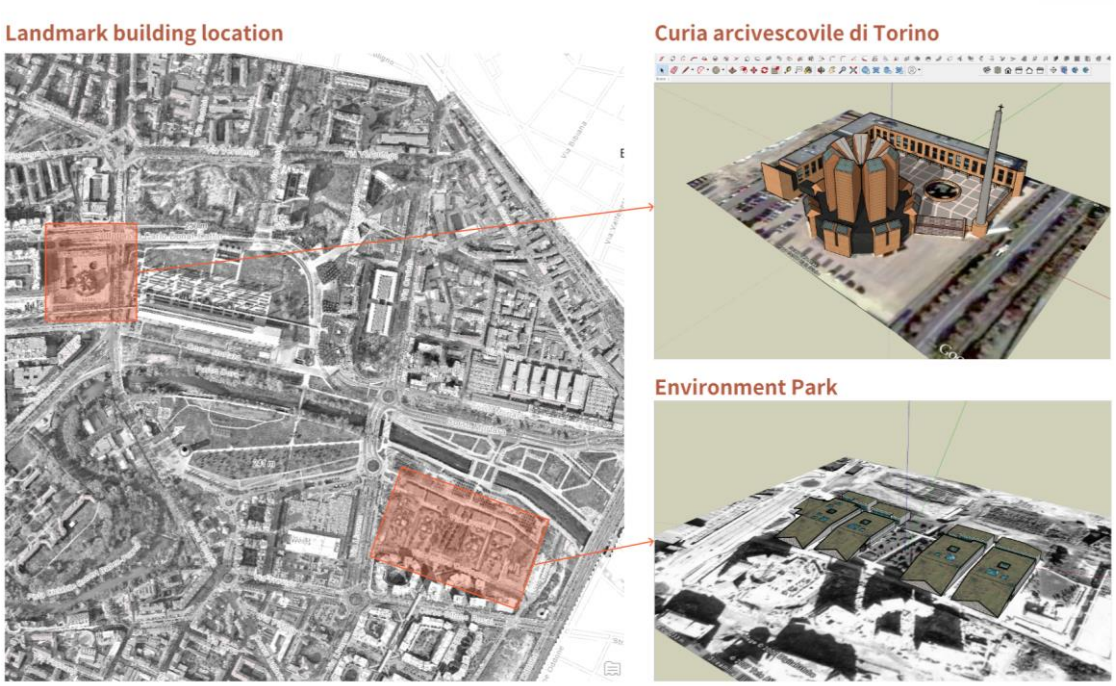


Figure 37. Location of Curia arcivescovile di Torino and Environment Park

- 4.2 Data Optimization in ArcGIS Pro

- 4.2.1 Orthophoto import and processing

Because the previously exported high-resolution orthophoto had a georeference, ArcGIS Pro automatically aligned it. Since the downloaded image covers a square extent larger than the actual study area, the Dora Park area boundary shapefile from Geoportale Piemonte was used to clip the TIFF image, ensuring that the aerial data precisely matches the spatial extent of the study area.
The step needs for this step is Toolboxes > Spatial Analyst Tools > Extract by Mask. Then, set the parameters as shown in the Figure 38:

Input raster	*dora.tif*
Input raster or feature mask data	*DORA_boundary*
Output raster	*DoraOrthophoto*

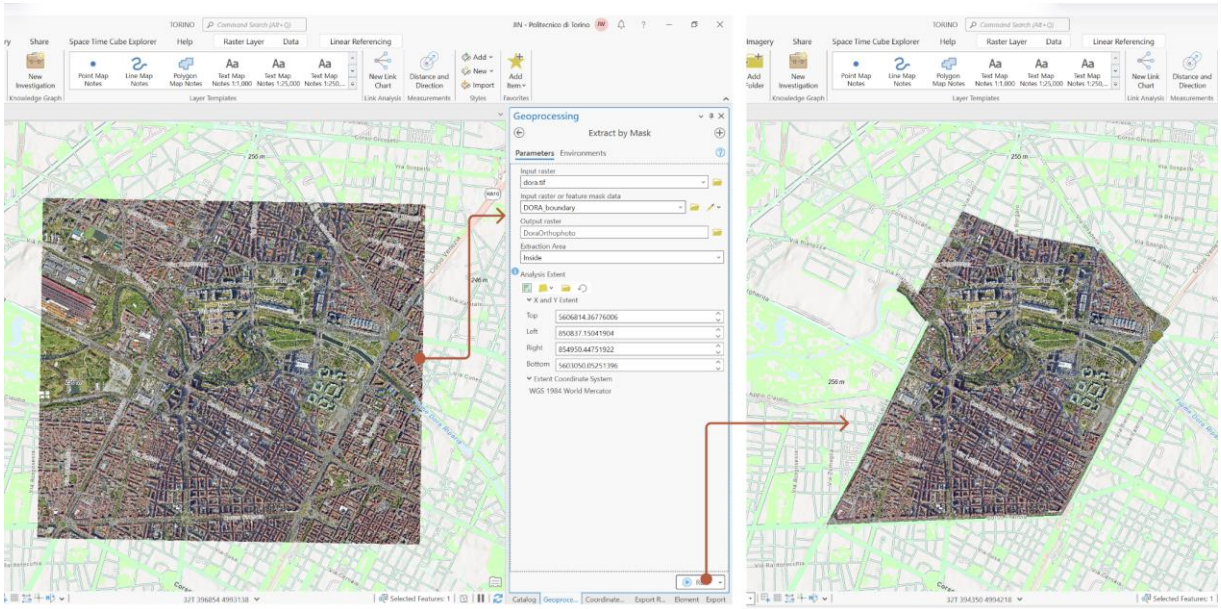


Figure 38. Parameter settings in Extract by Mask

- 4.2.2 DTM image import and processing

Coordinate system correction alignment:
Download the DTM image of the specified area from the Geoportale Piemonte platform. Its data attributes are

Data Type	*Raster*
Vertical Units	*Meter*
Cell Size	*5*
Format	*TIFF*

Pixel Type	*floating point*
Projected Coordinate System	*WGS 1984 UTM Zone 32N*
Projection	*Transverse Mercator*

The coordinate system is WGS 1984 UTM Zone 32N, the DTM image and the basemap are aligned.

Merging and clipping:

The two DTM images in TIF format, numbered 155 and 156, were imported into ArcGIS Pro. These two images share the same depth map, but their starting elevations are different. The elevation of the image on the right ranges from 159.51 to 715.4m, while the one on the left ranges from 220.47 to 2224.88m. To merge their boundaries, the Mosaic to New Raster tool from the Geoprocessing section was used. To ensure consistency with the raster properties of the original data, the pixel type was set to 32-bit float, the number of bands to 1, and the spatial reference for the raster to WGS_1984_UTM_Zone_32N. Finally, the images were mosaicked together, resulting in the pattern shown in Figure 39.

Figure 39. The result of stitching two TIF images

Finally, use the previously mentioned cropping tool (Extract by Mask) to segment the DTM image and only retain the area we need (Figure 40) .

Figure 40. DTM image of the DORA Park area

- 4.2.3 Vector data import and processing

Since the acquired vector datasets of buildings, rivers, and roads cover the entire city of Turin, the amount of data is substantial. To reduce computational load during subsequent processing and to ensure consistency with the case study area, the existing vector layers were segmented to retain only the features within the defined boundary.

This operation was carried out using the Clip tool in ArcGIS Pro, where the boundary of the Dora Park area was applied as the clipping feature to extract the relevant elements (Figure 41).

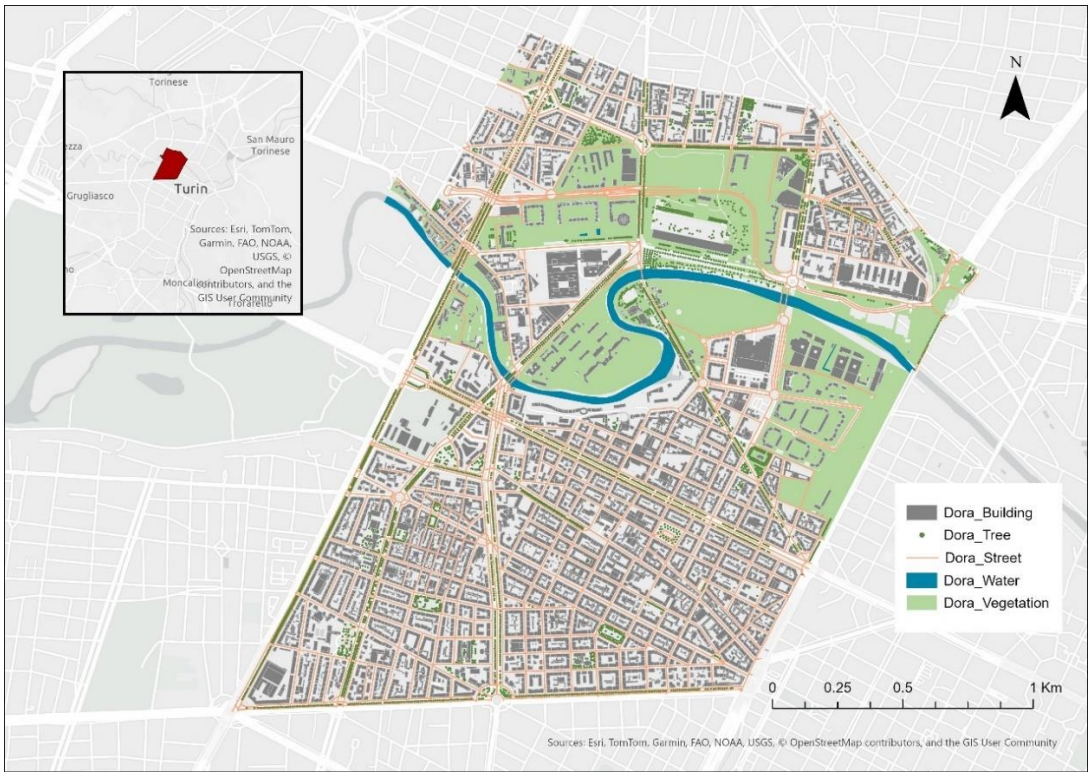


Figure 41. The resulting vector data obtained through Clip tool contain only the spatial features located within the study area.

- 4.3 Parametric modeling in CityEngine

- 4.3.1 Fundamentals of CityEngine Modeling

Shape Grammar:

The fundamental principle of modeling in CityEngine is essentially to define and program a shape grammar that drives the computer to complete architectural modeling. In practice, if there are too many rules, the use of shape grammar may become inconvenient. Taking famous architect Wright's Prairie Houses as an example, although the operation of shape grammar rigorously defined the category of such buildings, Koning and Eisenberg employed nearly one hundred rules to complete the grammar of this architecture (Figure 42)[37]. The architect's thinking also plays a crucial role in grammar construction and architectural design, as it requires the designer to think clearly about their creation and to express their work through precise grammatical relations.

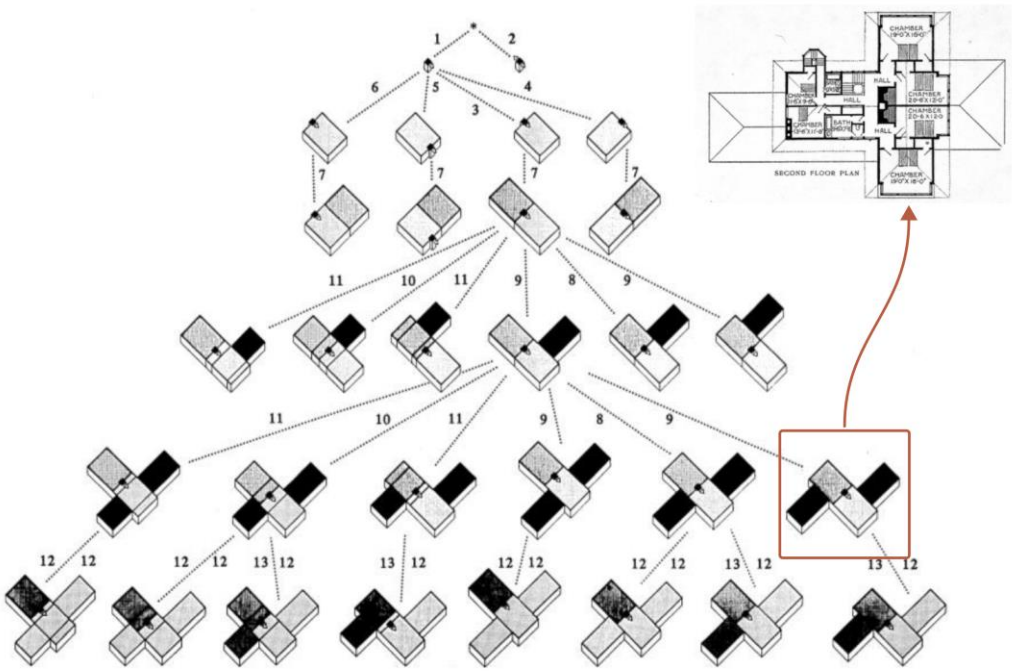


Figure 42. A section of the rule tree shows the valid sequences of shape-rule schema applications used to create basic compositions. The numbers shown on the branches match the specific schemata used at each step. Black-toned areas are functionally undistinguished zones, Wright's tendency to group services together so that they do not interrupt living zones.
Source: H Koning et al., *The language of the prairie: Frank Lloyd Wright's prairie houses*. 1981

Antony Radford and Dean Bruton stated:
"Merely presenting examples of past works is insufficient; one should begin with their language, vocabulary, and rules. Indeed, for other artists, a collection of past artworks is not as useful as a collection of past grammars. Innovation arises from understanding the boundaries of the past and transcending them [38]." The advantage of shape grammar lies in the fact that if the designer defines clear initial forms and grammar rules, then within this framework, the rules can generate all possible design outcomes. What the designer needs to do is filter and evaluate these potential results, and further refine those that meet the requirements. This significantly reduces the workload of searching for various design possibilities. When combined with computer programs, this method greatly expands the range of design possibilities and dramatically enhances design efficiency.

CGA Rule-Based Modeling:
CGA shape grammar is a modeling language unique to the CityEngine platform, capable of generating high-quality three-dimensional building models. By defining four components—shapes, attributes, operations, and grammar rules—it can construct various forms. In CGA shape grammar, shapes consist of symbols, geometry, and numerical attributes, usually recognized by symbols. The geometric attributes correspond to scope, which is a bounding box in space, and the most important geometric attributes are the position (P), the three orthogonal vectors describing the coordinate system, and the size vector (S). Shape operations are a very important component of shape grammar, mainly including four types: first, scope operations, which can modify the scope of a given shape,

including extrusion, translation, rotation, and scaling; second, split operations, which divide the scope along a given axis with specified dimensions as attributes; third, repeat operations, which replicate geometries in a given direction, and in CGA grammar they are often written as part of split rules; and finally, component split operations, which divide three-dimensional scopes into smaller shapes such as faces, boundaries, and vertices. Grammar rules iteratively modify and replace shapes, progressively refining the model by adding details such as walls, floors, windows, and doors. This hierarchical modeling process-from basic volume to detailed building components-is highly compatible with the concept of LOD commonly used in architectural and urban modeling. Each stage of rule application corresponds to a higher level of detail: the initial volume generation can be analogous to LOD 1; Definitions of structure and elevation elements correspond to LOD 2-3; Finer components containing openings, material properties, etc. approach LOD 4 (Figure 43)[39]. As shown in Figure 44, the generative modeling process based on CGA not only realizes the automatic generation of geometry, but also reflects the basic principle of LOD classification and refinement in digital city modeling. In this thesis, considering that the focus is on the research of modeling methodology and the efficiency of loading models, LOD2 is used for most buildings, and LOD3 models with more details are imported from outside only for important landmark buildings.

Figure 43. The five LODs gradually increase in geometric detail and semantic complexity, with LOD4 ultimately containing indoor features.

Figure 44. Iterative evolution of CGA modeling from simple to complex

The CGA rule-based modeling method is a typical digital modeling approach, with its rule expressions resembling the textual descriptions of historical models. It precisely defines the process by which a simple two-dimensional shape evolves step by step into a complex three-dimensional model. Unlike the real world: although both virtual and real architectural construction follow the principle of evolving from simple to complex, in the real world construction proceeds from the skeleton outward, while virtual three-dimensional modeling proceeds from large blocks subdivided into smaller volumes—from large to small. In the real world, construction is an additive process of materials, whereas in the digital environment construction can be additive or subtractive, representing a constantly evolving and refining process.

Pre-installed rules are available within the ESRI.lib project. Additional high-quality rule sets suited to various scenarios can be accessed through the tutorials and downloads section (Help → Download Tutorials and Examples). Moreover, a wide range of community-created rule packages (.RPK files containing rules and related assets) can be found online by searching for “ArcGIS content”

together with the keyword CityEngine (Esri, 2019c). Examining these existing rules provides valuable insight into how models are generated using the CGA language. Since developing custom rules can be time-consuming, it is recommended to reuse available libraries and rule sets whenever possible before writing new CGA code [40].

To apply a rule or rule pack, drag the desired file directly from Navigator onto a shape, as shown in Figure 45. If more than one shape is selected, the rule applies to all objects at once.

The Inspector panel provides several options for modifying rule parameters. Users can also use the right-click menu to select shapes by layer, set start rules, and so on. When a rule is assigned, the system compiles and calculates briefly to generate the corresponding model. In order to obtain finer control, Inspector also provides detailed settings such as CGA rule file path, starting rules and related attributes.

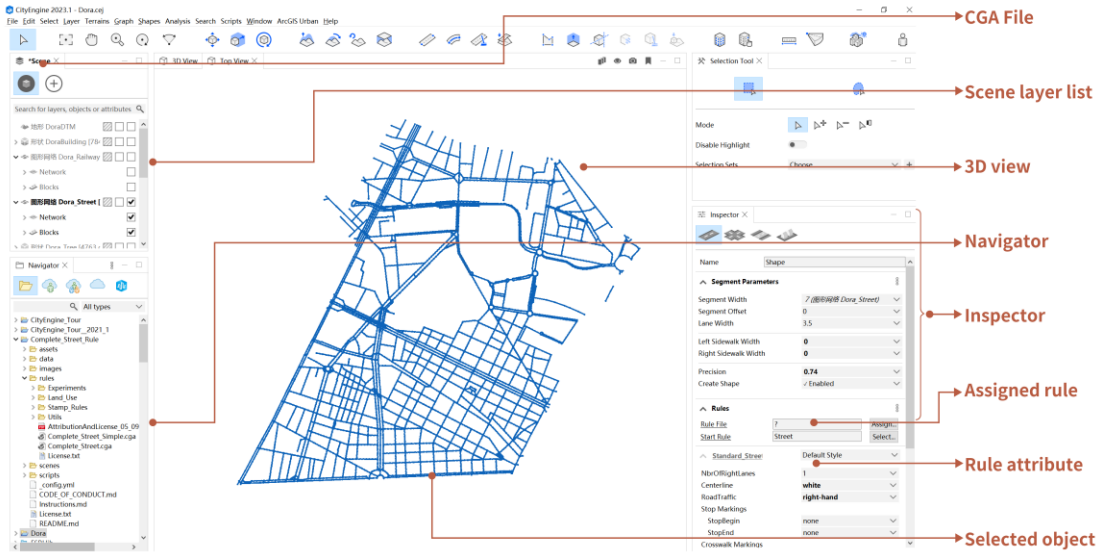


Figure 45. CityEngine user interface elements. Orange: important elements of the interface. Blue: dragging a rule onto the selected shape to generate a 3D model

Structurally, CGA rules are somewhat similar to architectural shape grammars, which have long been used in architectural theory to describe building structures or analyze architectural forms (Table 4). With the development of powerful tools such as CityEngine, architectural grammar has gained an interesting platform for practical application.

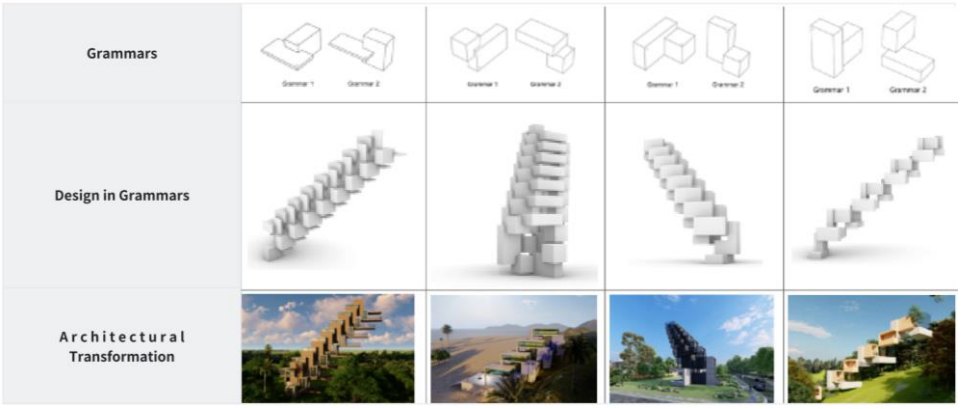


Table 4. Application of Shape Grammar in Architectural Forms

Source: *Ranran Wei et al., A Study of the Influence of Shape Grammar on Architectural Form.*2021

Efficiency of CGA Modeling

In terms of modeling efficiency, traditional manual modeling still holds significant advantages when applied to single buildings or small scenes. However, when large-scale scenes need to be modeled, the workload of traditional modeling often increases proportionally with the scale of the scene, making manual modeling inadequate. CGA rule-based modeling belongs to parametric modeling. Once the necessary rules are written according to requirements, the remaining process is handled by the computer. The differences in cost and efficiency between the two approaches are illustrated in Figure 46.

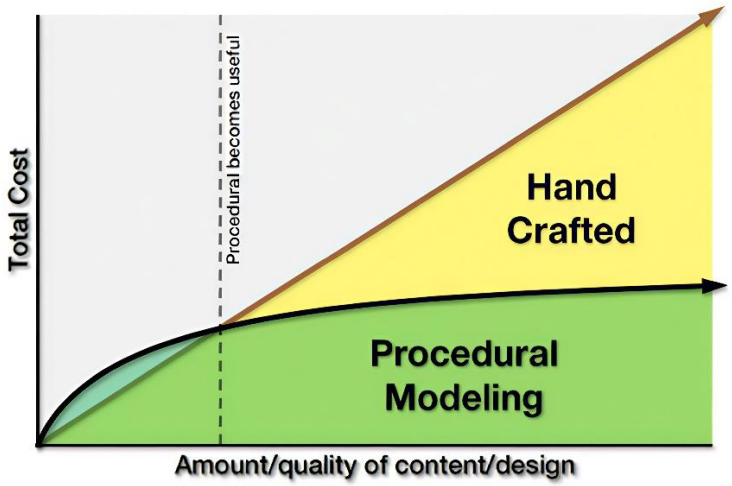


Figure 46. Comparison of manual modeling and programmatic modeling costs

In the field of urban planning, roads are regarded as the skeleton of a city, while buildings are the primary artificial structures that accommodate citizens' work and living activities, serving as the main carriers of the city's spatial image. Therefore, the core of three-dimensional visualization in a digital city lies in the visualization of roads and buildings as the fundamental urban elements. Given the unique advantages of parametric technology in three-dimensional modeling, particularly in large-scale modeling, this study takes roads and buildings as examples to explore methodology of 3D modeling using a parametric platform.

- 4.3.2 The structure of CGA rule grammar

A complete set of CGA rule grammar generally consists of four components: rules and custom functions

Different type of CGA rules:

According to their different functions in the model generation process, CGA rules can be classified into standard rules, parameterized rules, random rules, conditional rules, and recursive rules.

1. Standard Rules

The function of standard rules is to return the result of the operation and to identify the generated

model.

The general syntax of this type of rule is as follows:

PredecessorShape --> Successor

Example Rule:

Lot-->extrude(20) Mass

This rule extrudes the lot shape by 20 units to generate a building mass (Figure 47).

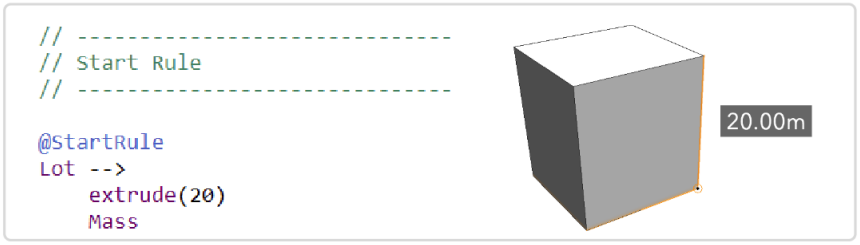


Figure 47. Shown in CityEngine

2. Parameterized Rules

Parameterized rules are similar to functions with parameters. Their main characteristic lies in the inheritance of models, enabling the transmission of modified appearance information of the successor model as required. In other words, modifications to the successor model can be achieved through the adjustment of parameters in the predecessor model.

The general syntax of this type of rule is:

PredecessorShape(Parameters) --> Successor

Among CGA rules, parameterized rules are the most commonly used. However, during their application, attention must be paid to the types of parameters. Parameter types are not strictly limited—they may include Boolean (bool), floating-point numbers (float), strings (string), or even expressions. Additionally, parameterized rules with the same prefix but different numbers of parameters are analogous to function overloading; in CityEngine, such rules are recognized as distinct rules.

Example Rule:

Lot -->Mass(20)

Mass(height)-->extrude(height) Building

This rule extrudes the mass by the given height to generate a building (Figure 48).

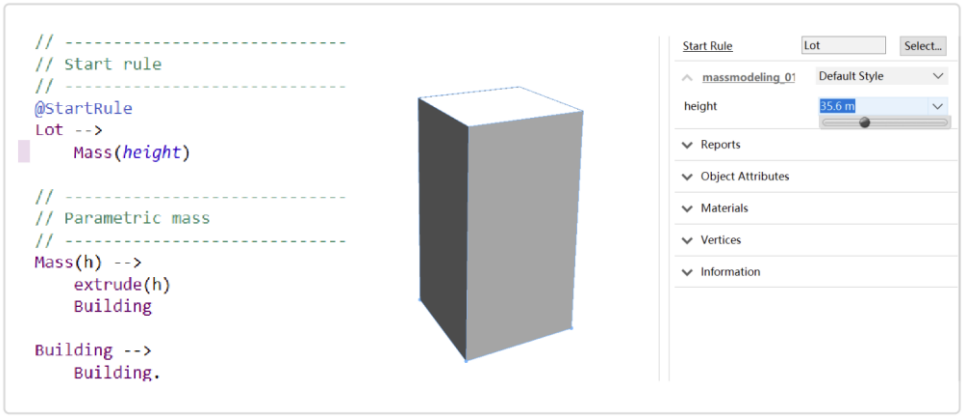


Figure 48. Shown in CityEngine

3. Random Rules

Random rules are used to generate models with varying appearances in a stochastic manner. Their structural form is as follows:

PredecessorShape -->

percentage%: Successor1

percentage%: Successor2

...

else: SuccessorN

During their usage, it should be noted that no other statements may appear outside the random statement, and the rule must terminate with an else clause. Furthermore, the sum of percentages before else must not exceed 100.

Example Rule:

```
Lot -->
50%:Mass(10)
20%:Mass(20)
else:Mass(30)
Mass(height)-->extrude(height) Building.
```

This rule randomly generates buildings of different heights (10, 20, or 30 units) on the lot, then extrudes them into building volumes (Figure 49). However, in this study, the actual building height is linked to the height attribute in the building's GIS data to generate the true height.

```
@StartRule
Lot -->
    HeightRandom

// The building height is randomly selected based on probability.
HeightRandom -->
    50%: Mass(10)
    20%: Mass(20)
    else: Mass(30)

// Parameterized Mass Rules
Mass(h) -->
    extrude(h)
    Building

Building -->
    Building.
```

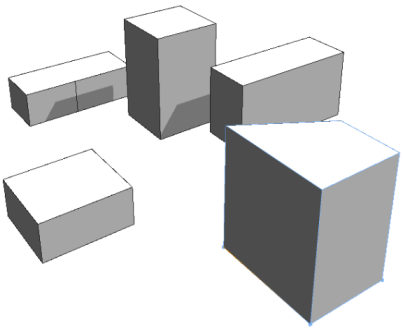


Figure 49. Shown in CityEngine

4. Conditional Rules

Conditional rules are realized by using the keywords case and else, and their process is similar to conditional statements in functions. They enable the generation of different model appearances depending on the specified conditions. No other statements may appear outside the conditional structure. The syntax is expressed as follows:

PredecessorShape -->

case condition1: Successor1

case condition2: Successor2

...

else: SuccessorN

Example Rule:

Lot -->

case geometry.area<1000:Mass(10)

case geometry.area>2000: Mass(20)

else: Mass(30)

Mass(height)-->

extrude(height)

Building.

The rule generates building masses of different heights based on the plot area: smaller plots correspond to lower buildings, while larger plots generate taller buildings (Figure 50).

```
@StartRule
Lot -->
    case geometry.area < 1000 :
        Mass(10)
    case geometry.area > 2000 :
        Mass(20)
    else :
        Mass(30)

// ---------------------------------------
// Mass extrusion rule
// ---------------------------------------
Mass(height) -->
    extrude(height)
    Building

Building -->
    Building.
```

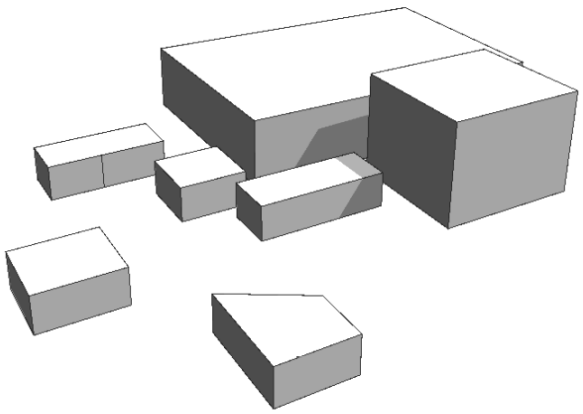


Figure 50. Shown in CityEngine

5. Recursive Rules

The principle of recursive rules is to use iterative statements that enable the model to repeatedly execute a specific instruction under certain conditions.

Form:

PredecessorShape -->

case condition_1: Operations PredecessorShape

case condition_2: Successor2

...

else: SuccessorN

The characteristic of recursive rules is their similarity to loop statements in functions. They are applicable in scenarios where the same operation needs to be executed repeatedly when certain conditions are satisfied.

Example Rule:

Lot -->

extrude (10)

comp(f) {side:SideFacade | top:Recursive}

Recursive-->

case geometry.area>100:

Z.

s(5,5,1)

extrude (5)
center(xz)
comp(f) {side:SideFacade | top:Recursive}
else:
Z

This rule creates a red building mass where the top face is recursively extruded and reduced until the area becomes small, simulating a stepped or terraced form (Figure 51).

```
@StartRule
Lot -->
    extrude(10)
    comp(f){ side: SideFacade | top: Recursive }


// -------------------------------------
// Recursive Rule
// -------------------------------------
Recursive -->
    case geometry.area > 100 :
        Z
        s(5, 5, 1)
        extrude(5)
        center(xz)
        comp(f){ side: SideFacade | top: Recursive }
    else :
        Z
```

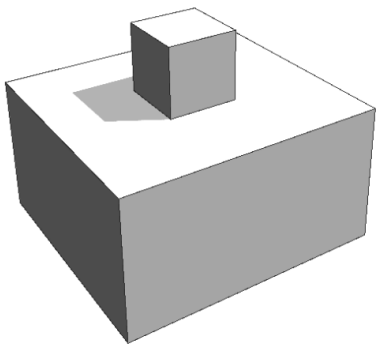


Figure 51. Shown in CityEngine

6. Custom Rules

Custom functions in CityEngine are similar to attributes. They can be parameterized, randomized, and conditioned.

Example Rule:

attr Floor=0
height=
case Floor<2:5
case Floor<6:4+3.2*(Floor-1)
else:3.8+3*(Floor-1)
Lot -->
color(1,0,0)
extrude(height)

The rule dynamically calculates the building height based on the number of floors (Floor) and extrudes the lot (Lot) into a volumetric building mass with the corresponding height. By applying conditional statements, the code differentiates between lower floors (1st floor), middle floors (2nd–5th floors), and upper floors (6th floor and above), thereby establishing a parametric modeling logic that more accurately reflects real-world architectural practices (Figure 52).

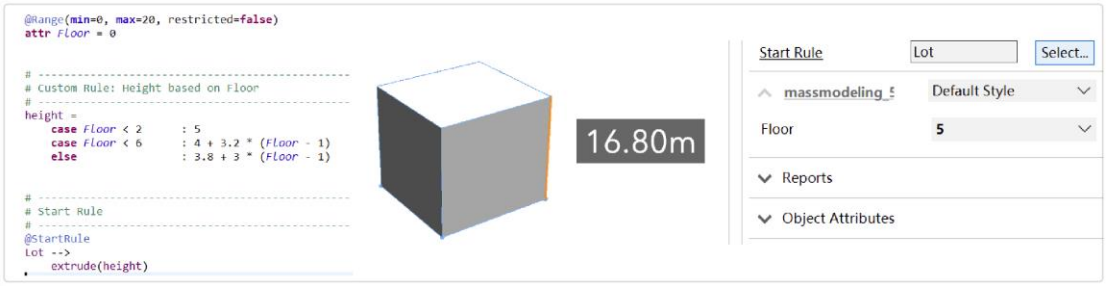


Figure 52. Shown in CityEngine

- 4.3.3 Terrain generation

Principles and Methods

Terrain modeling in CityEngine aims to import, construct and refine the landform of the target area, and provide a spatial basis for urban design and 3D visualization. The core idea is to represent relief using raster data such as DEM, which records continuous elevation values through a regular grid. This elevation data can then be integrated with vector layers such as road networks and building outlines to build a unified and coherent 3D scene.

CityEngine mainly uses two steps to build terrain:

1. Data-based terrain construction:

Importing DEM or raster grids (such as GeoTIFF, ASCII grid, and so on) directly as terrain. Since these data have their own geographical references, the generated surface can be accurately aligned with urban GIS data, thus ensuring spatial consistency.

2. Terrain refinement:

After generating the topo surface, you can refine it by adding material layers (such as grass, water, rock) and using tools such as Terrain edit, smooth, reset brush.

For CityEngine, terrain is not only the basis for placing roads, vegetation and buildings, but also an analytical tool (such as visual analysis). This helps stakeholders to better understand how the proposed project and the natural landscape interact with each other in function.

Here's a breakdown of each part's key content:

1. Data Acquisition and Import

Elevation datasets (DEMs) are first collected from open geospatial databases or authoritive sources. These datasets are then imported into CityEngine, where coordinate systems are checked to ensure alignment with other project layers.

When it comes to file management, CityEngine uses a folder-based structure. The software has built-in folders like assets, data, maps, models, rules, scenes, and scripts—each meant to manage specific file types. In CityEngine, the maps folder (Figure 53) is specifically used for storing terrain-related files. The Parco Dora DEM and orthophoto TIFF files—previously processed in ArcGIS Pro—are placed into this folder, making subsequent import into the project more convenient.

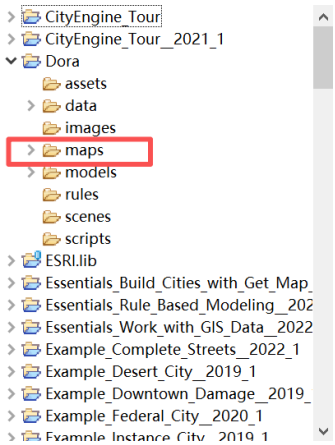


Figure 53. CityEngine folder management

2. Terrain Mesh Generation

CityEngine turns raster values into a 3D mesh, which can be further adjusted using grid resolution or level of detail. Finer resolutions result in more lifelike terrain but put greater strain on computing resources.

When generating terrain, the orthophoto imagery is first dragged into the workspace to serve as the base terrain. At this stage, the terrain already comes with a basic texture map. A DTM file is needed next to act as the elevation parameter—this creates a base map with lifelike topographic variations (Figure 54). This terrain will contain compressed building images, which will be overlaid on top after the subsequent building modeling is completed.

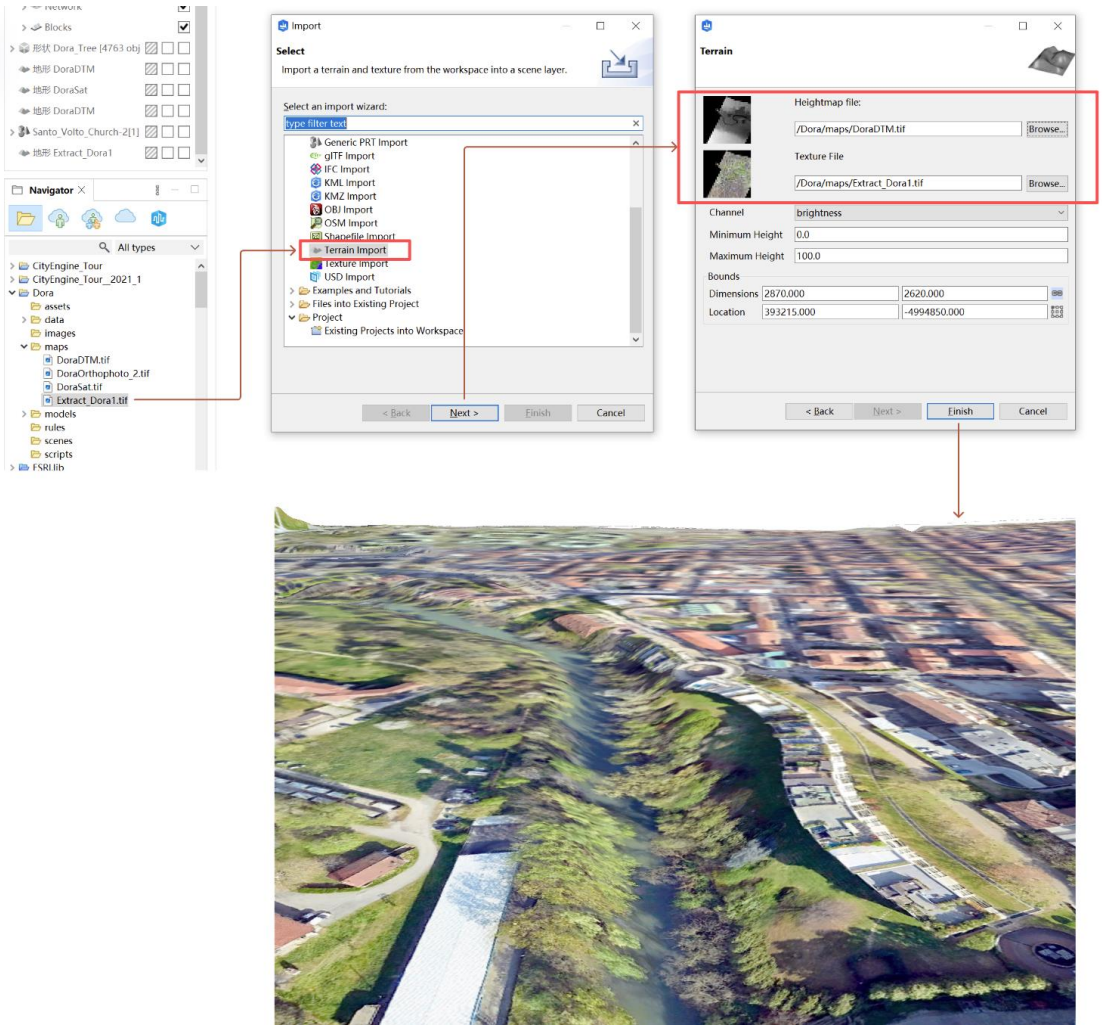


Figure 54. Textured terrain generation, the architectural images are compressed, and the completed architectural model will be overlaid on top.

3. Surface Texturing

Textures such as aerial imagery, land cover maps, or parametric materials are applied to improve visual quality. Usually, shp files of buildings, trees, rivers, etc. are imported to improve the realism of the model.

Import the river and vegetation shp file from the data folder (Figure 55).

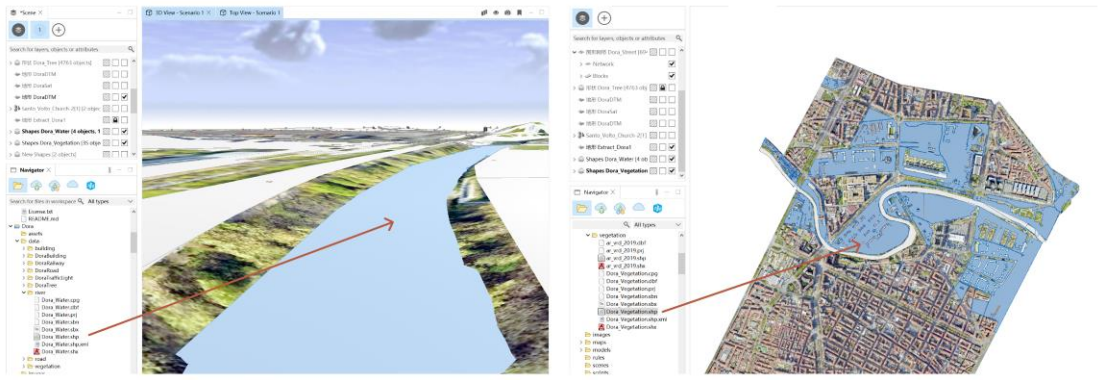


Figure 55. Dora Riparia river and vegetation generation

4. Terrain Adjustment

Local modifications can be made to prepare design sites—like flattening areas for construction or cutting slopes for roadways. Parametric rules can also recreate natural or man-made features, such as river channels or terraced platforms.

5. Integration with Urban Elements

When roads and buildings are modeled, urban features need to be further matched to the topographic surface: road networks are projected onto the terrain, buildings automatically fit to the surface based on elevation values, and vegetation is arranged according to slope or soil type. Through this process, spatial consistency and continuity can be maintained between the terrain and urban elements.

- 4.3.4 Roads modeling

Principles and Methods of Road Modeling:

The principle of road modeling is to decompose and classify the whole street system, and apply corresponding parameter settings to different components. By adjusting the parameter values, the design and modification of roads can be completed quickly and conveniently. When linear vector data is imported into CityEngine, the system automatically generates a road network based on preset parameters such as street width (streetWidth), left sidewalk width (sidewalkWidthLeft), right sidewalk width (sidewalkWidthRight), and so on. The resulting road geometry can be presented in a simplified form as shown in Figure 56, and different road structures will generate different initial forms according to default rules.

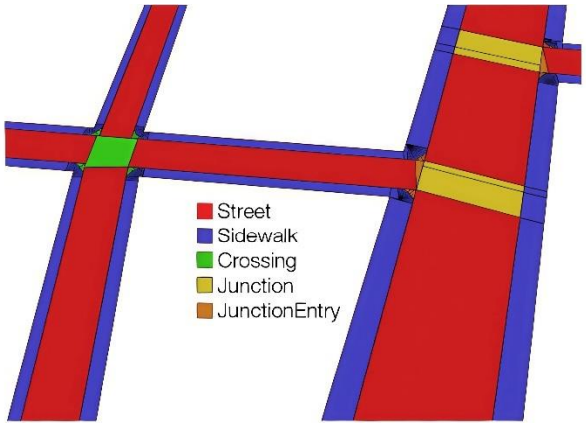


Figure 56. Default initial rules for road data

For instance, the green “Crossing” represents the default initial rule for the geometry of a road intersection. When the rule file contains a “Crossing” rule, the green portion will directly apply this rule. In CityEngine, when writing road rules, it is necessary to apply different initial rules to different components. The logic for writing the rules of Street, Sidewalk, Junction, and JunctionEntry is essentially the same as that for Crossing.

The basic design idea of parametric road modeling is as follows: along the direction of the road centerline, the street is subdivided, and further subdivision is carried out along the cross-sectional

direction of each element, disassembling the road into components whose parameters can be independently adjusted. Each component generates different models through shape symbols or texture images, thereby realizing three-dimensional modeling of roads.

The main contents of each part are:

1. Data Acquisition and Import

Import the road from the data folder into the operation interface and check whether it is aligned with the previous aerial orthophotos image. After importing, the software will give the road a basic width: total road width = 7 meters, each lane 3.5 meters (Figure 57)。

Figure 57. Road parameters in inspector

2. Roads correction

The road network generated in the initial import stage contains several inconsistencies when compared with the actual street layout. In some areas, the road widths do not match the real dimensions, and certain segments are misaligned or overly fragmented. These issues become particularly evident in the streets located below the church. As shown in Figure 58, several problems appear simultaneously: some roads are missing at intersections; in other cases, the shapefile contains overlapping or duplicated line segments that produce unnecessary branches; and parts of the network include small residual segments that create irregular blocks.

Since these errors originate from the underlying vector data quality, they must be corrected manually in CityEngine. This study used the software's basic editing tools to accomplish adjustments such as moving road centerlines, redrawing missing segments, and refining road curves based on geometry in the orthophoto. Through these manual clearing steps, the road network used can more accurately reflect the real world space conditions.

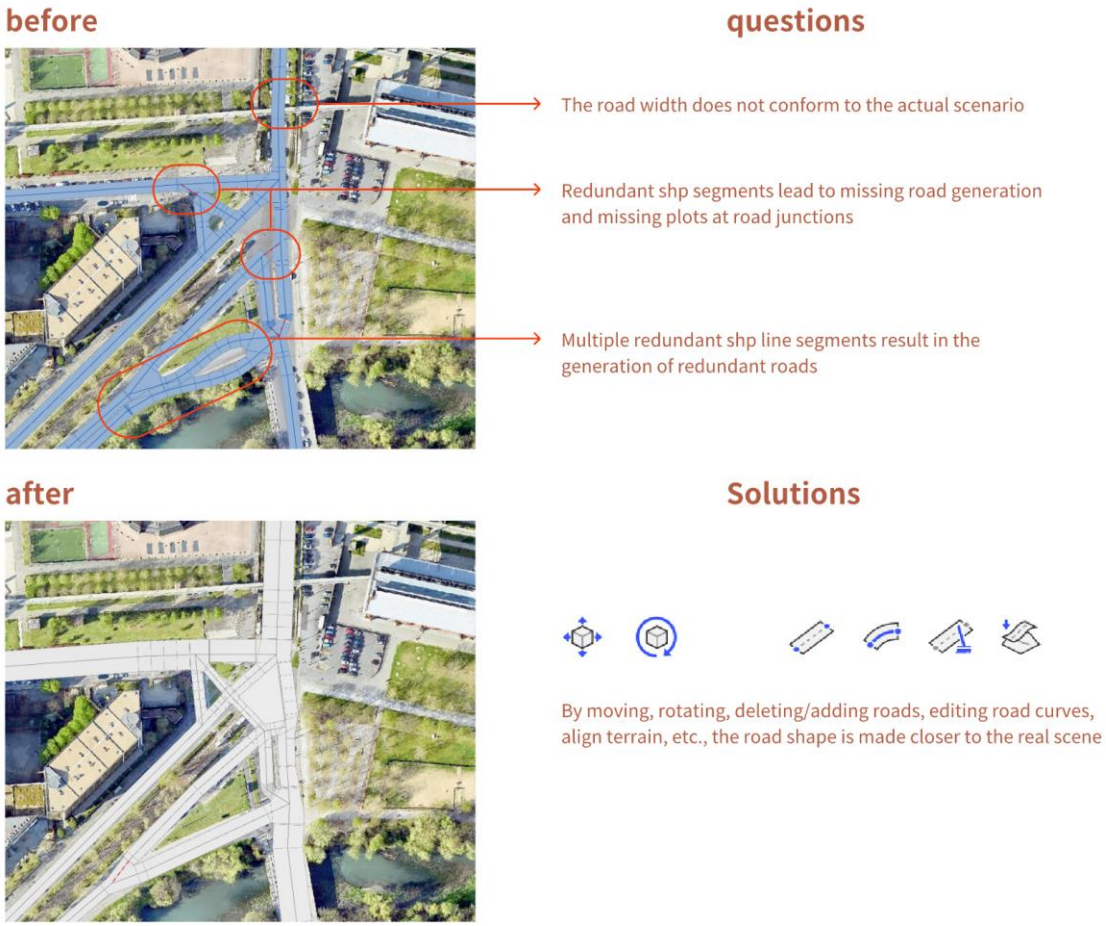


Figure 58. Changes before and after road editing

3. Refine modeling according to real scenarios

After the initial corrections to the road network, the street geometry still required further refinement so that the modeled sections would reflect actual conditions on site. To accomplish this, appropriate CGA rules were assigned to the network, enabling detailed adjustments of each cross-section. In this work, the refinement process drew on the "Complete Street Rule" published within the ESRI community[41].

This rule set provides a context-based framework to help users parametrically generate multi-modal street systems in the City Engine, taking into account a variety of commonly used traffic design codes, including NACTO guidelines, AASHTO standards, and MUTCD guidelines. The goal of this rule is to support the construction of 3D street types that can be compared and evaluated during the planning and design phases.

Although this rule is often used as a quick visualization tool, it also contains a set of dynamic indicators that automatically update as the geometry or parameters of the street change. These responsive metrics demonstrate how programmatic modeling can link design choices with real-time assessment and visualization, both to facilitate rapid testing of different street configurations and to make the presentation of planning options clearer and easier to understand.

Complete Street Rule covers a wide range of urban and suburban street types-including lanes with shoulders, road corridors with physical barriers, or sections containing tram tracks-and is fully integrated into the CityEngine's modeling environment. The generated streets can be exported to various 3D formats, or published as scene layers to online platforms, and can also be imported into real-time engines such as Unity or Unreal for immersive display and public participation.

Like Figure 59, the street located at the southeast corner of the church provides one example. The widths of the carriageway, the planted strips, and the tram tracks were measured manually from Google Earth, the data from these measurements are inaccurate because only for testing or demonstrating the procedure. These measurements were then entered into the CityEngine Inspector, where the corresponding parameters were adjusted until the modeled cross-section matched the real-world configuration.

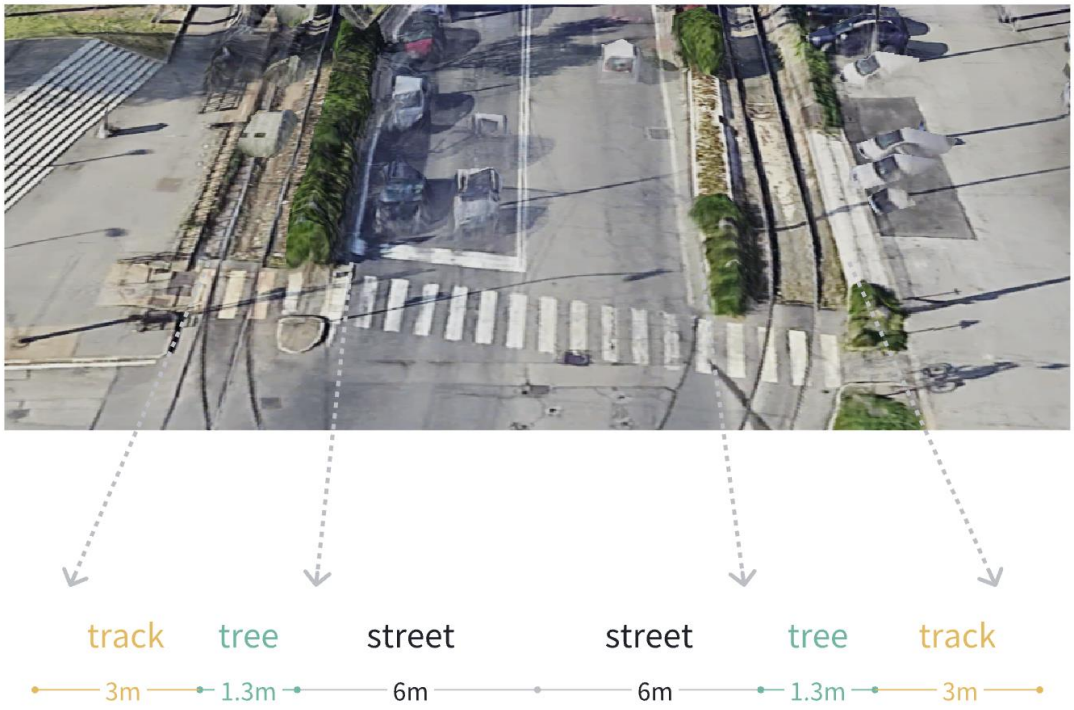


Figure 59a. Widths of different components of east road section used for modeling

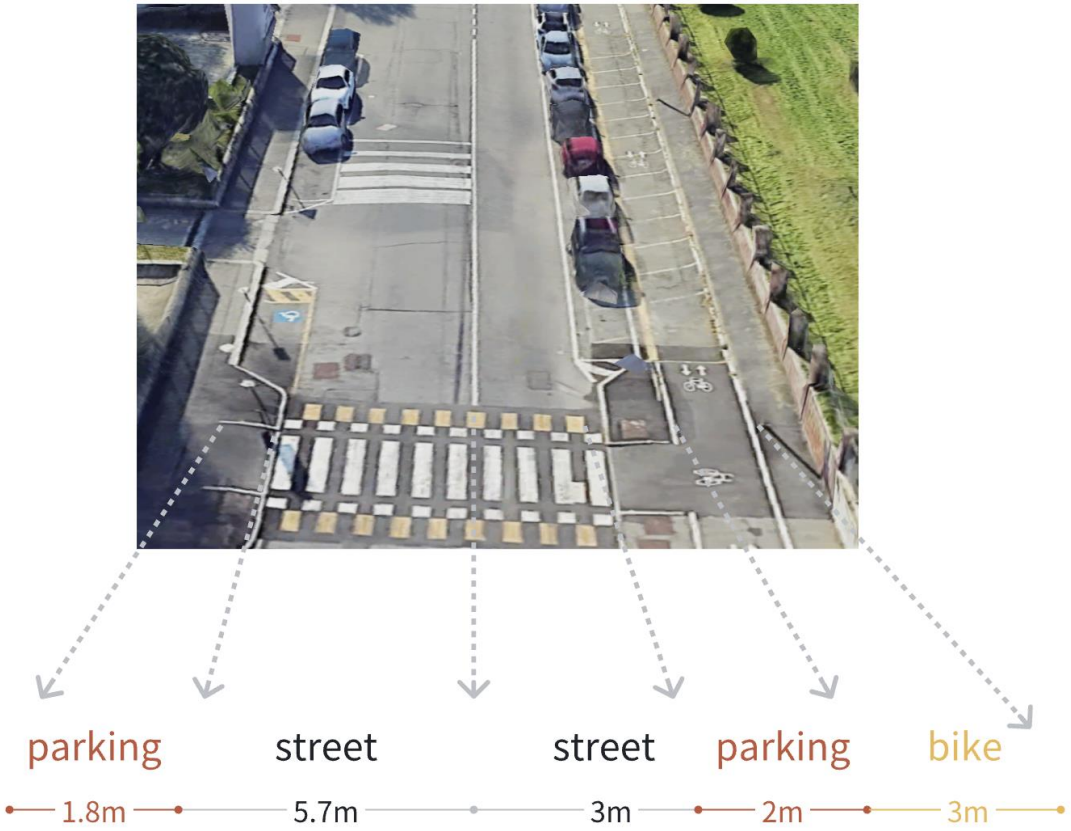


Figure 59b. Widths of different components of south road section used for modeling

According to the road section setting parameters, it is necessary to follow a certain logic: road total width planning, road layout, road center section layout, and other functional roads. In the specific rule-design process, the textures to be used must first be defined, including those for roadways, sidewalks, and intersections. For example, defining a street texture. Road facilities used in modeling, such as lampposts and street trees, can be imported for use, and must also be defined. The following Figure 60 shows the entire process of 2 roads modeling.

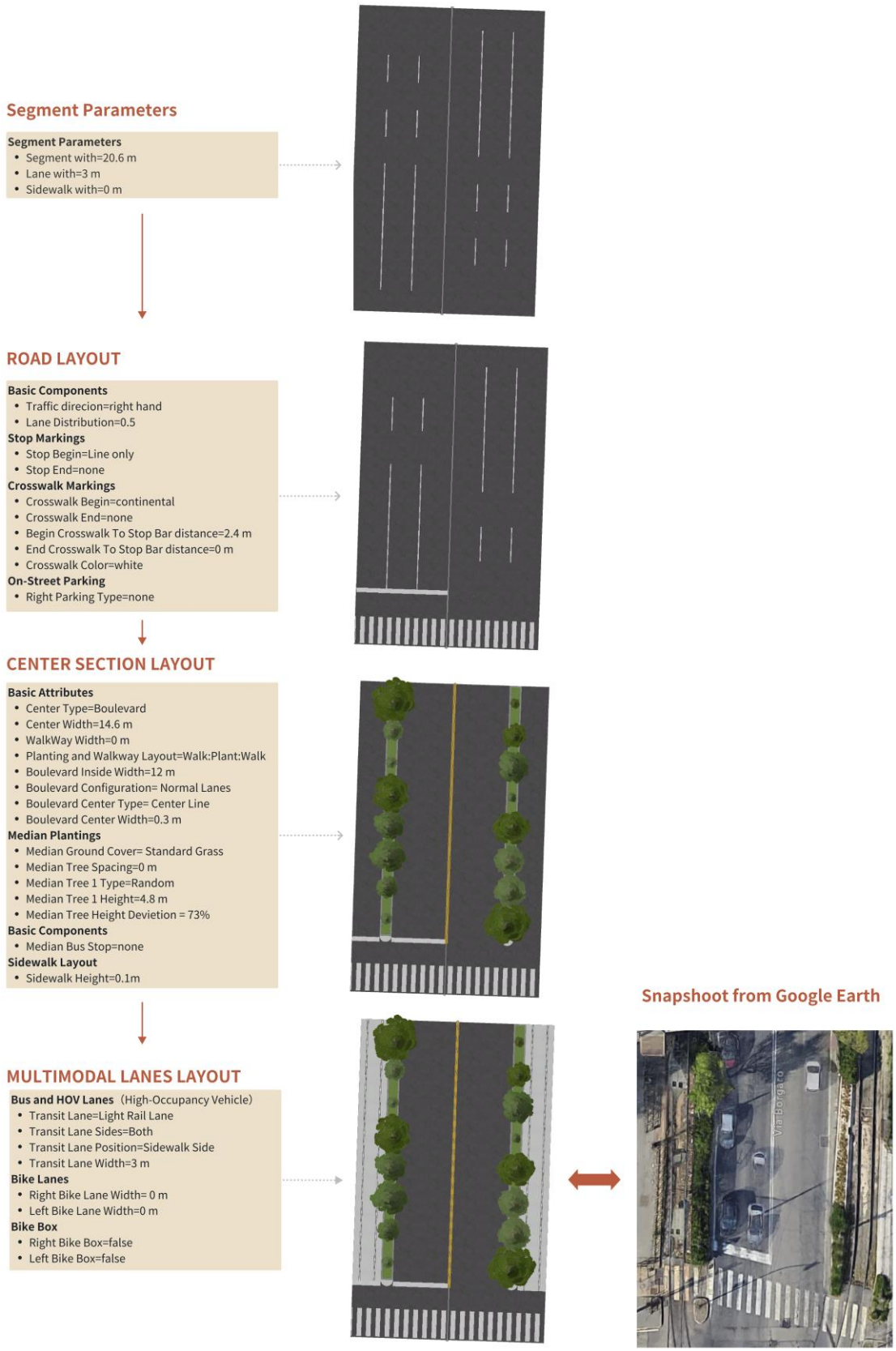


Figure 60a. Parametric east road modeling process

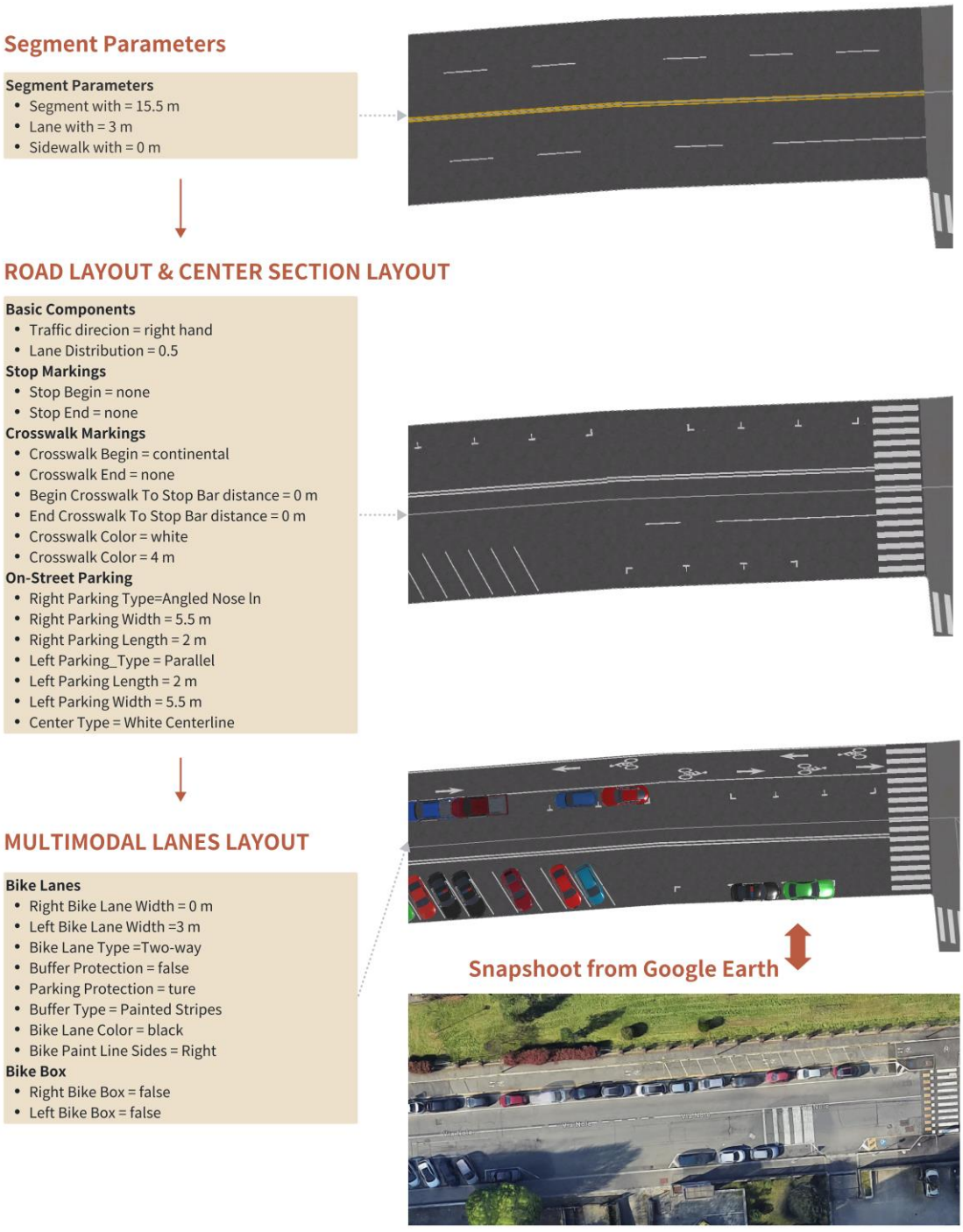


Figure 60b. Parametric south road modeling process

In summary, the workflow above shows how the road network was reconstructed in CityEngine by bringing in the original shapefile data, correcting its geometry, and then assigning suitable parametric rules. Using the Complete Street Rule made it possible to rebuild the main components of each cross-section—such as the carriageway, sidewalks, and planted strips—with proportions closer to the actual site. Several parts of the network still required manual intervention, particularly where widths had to be adjusted or redundant line segments removed, so that the layout matched the real configuration visible in the top-view reference (Figure 61). Although the procedure involved

both automated and manual steps, it proved effective for producing a coherent and adaptable 3D street model that can serve as a reliable basis for subsequent analysis and visualization.

Figure 61. Top View of the modeled roads at the southeast corner of the church

- 4.3.5 Buildings modeling

In cities, apart from some larger public buildings, most architectural forms are mainly composed of basic geometric shapes (such as squares, rectangles, etc.). Since the modeling design of more complex buildings falls within the domain of architectural design, such models will be obtained from the previously mentioned 3D Warehouse. In this thesis, only the modeling of buildings with relatively simple forms is discussed. For such buildings, the basic workflow of creating a single model is as follows: extrude the planar shape into a three-dimensional form, subdivide the surface of the form into floors, further subdivide each floor into functional areas such as windows and doors, add corresponding components to the functional areas, and finally apply materials（Figure 62）.

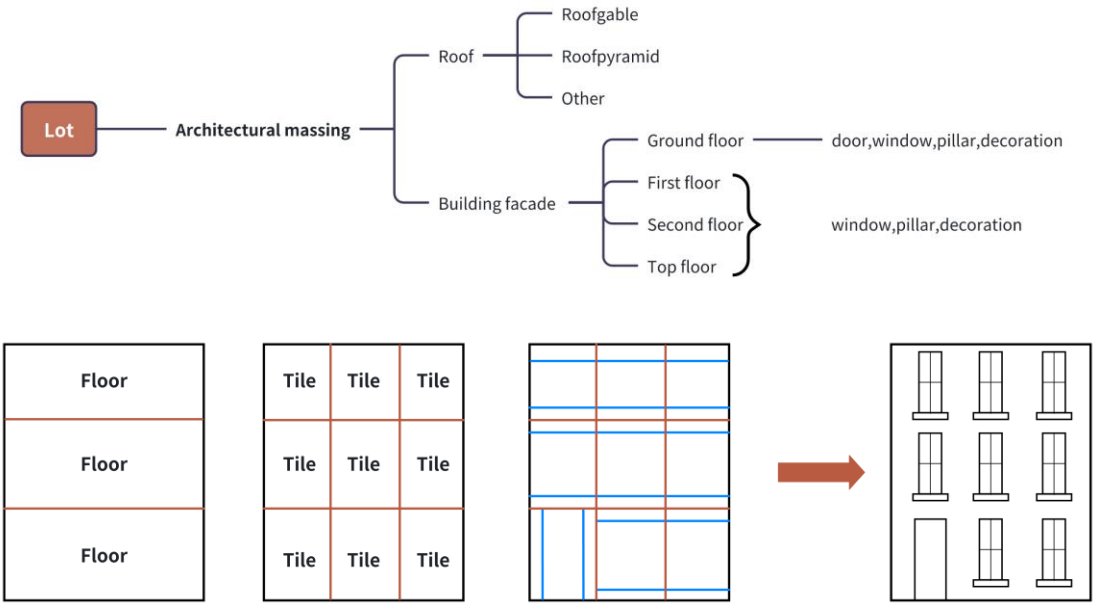


Figure 62. Basic process of architectural modeling

For relatively fine and complex components, such as windows, other modeling software can be used to create the components, which are then uniformly imported according to predefined rules and positioned accurately using coordinate adjustments, thereby achieving rapid and precise results. The following Figure 63 illustrates the roof component (.obj file) contained within the rule file assets.

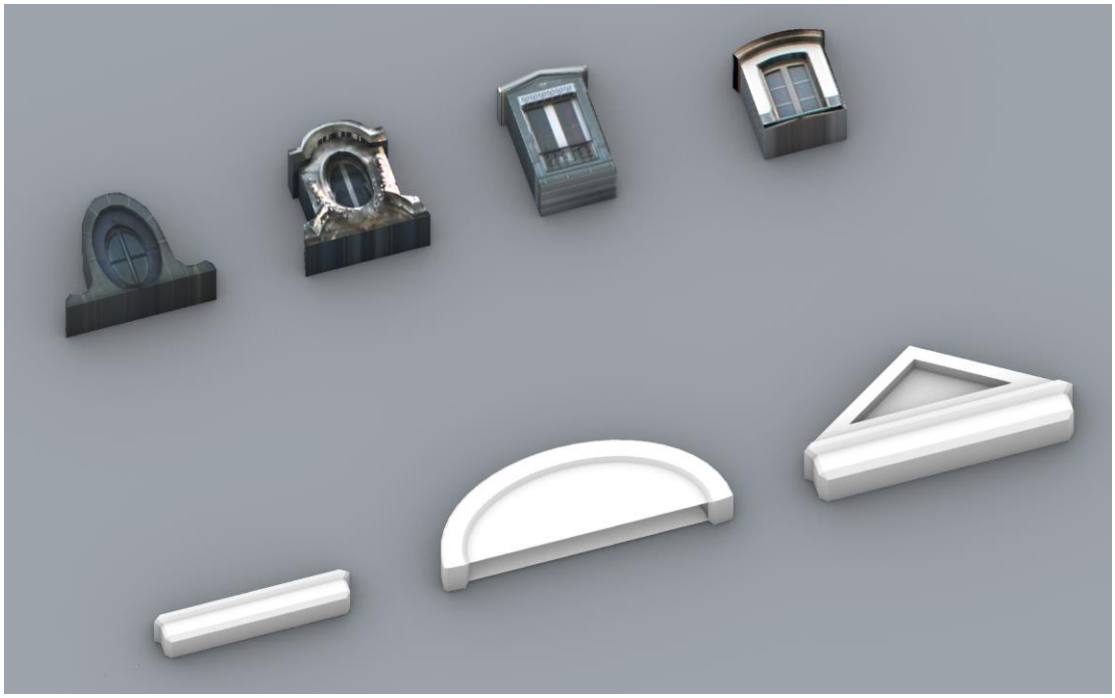


Figure 63. The .obj files for the attic (above) and the balcony (below).

In addition to structural components, materials also play a crucial role in expressing architectural characteristics. A large number of default texture maps—such as those for facades, roofs, doors,

windows, and decorative elements—need to be prepared. These textures are invoked by CGA rules and applied to the buildings accordingly. The roof textures are shown in Figure 64a, while the building facade textures are illustrated in Figure 64b.

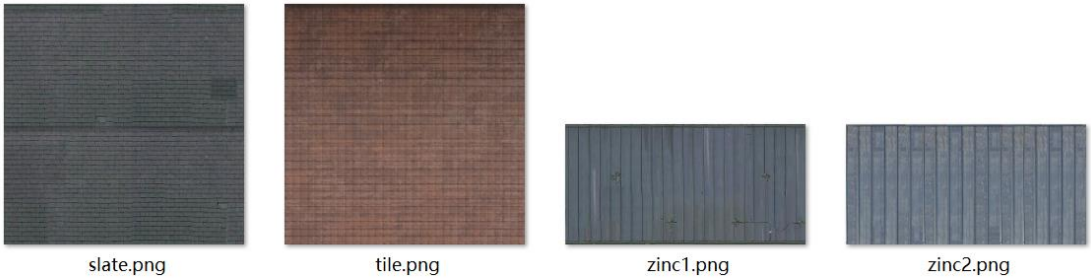


Figure 64a. roof textures

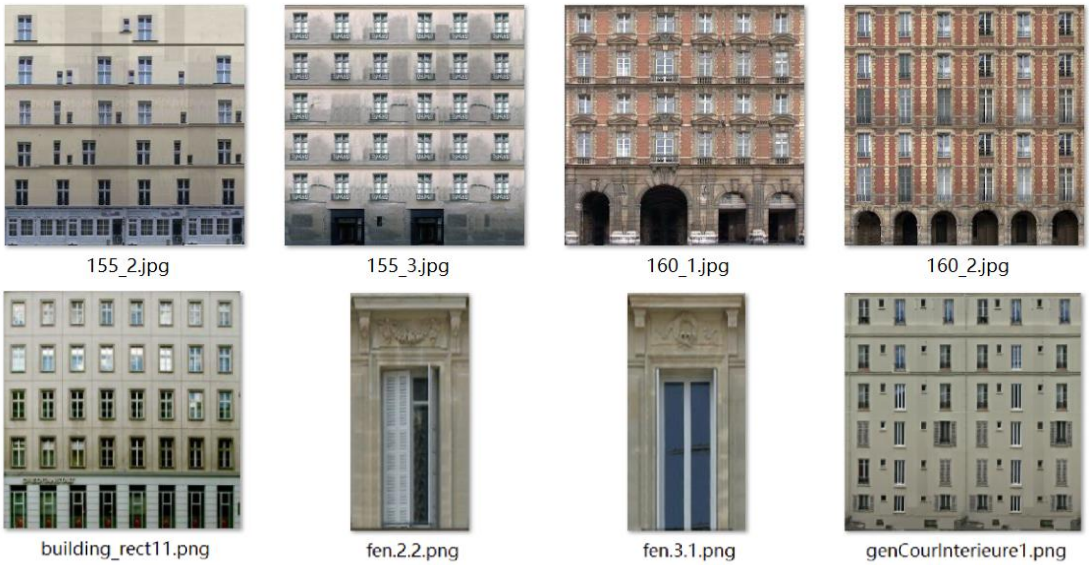


Figure 64b. building facade textures

The building modeling process follows a series of straightforward steps.

1. The building footprint shapefile was linked to a rule set taken from the ESRI rule library. A rule that reflects Turin's typical historic fabric was selected, as it allows the generation of façades with continuous balconies, pitched roofs and similar architectural facade styles.

2. Several attribute fields were introduced (Figure 65), enabling the building parameters to be edited directly through the Inspector panel. These attributes—such as LOD, construction year, overall height, ground-floor height, floor height, and roof pitch—provide the numerical basis for extruding each footprint to an appropriate elevation.

```
# --------------------------------
# Building Parameters
# --------------------------------

@Group("Building Parameters")

@Range(0, 1)
attr High_LoD = 1                    # Level of detail switch (0 = low, 1 = high)

@Range(1600, 2000)
attr Year = 1900                     # Building construction year (influences style and materials)

@Range(5, 100)
attr Height = 15                     # Overall building height

@Range(2, 10)
attr Groundfloor_Height = 4          # Ground floor height

@Range(2, 6)
attr Floor_Height = 3                # Height of each typical floor

@Range(1, 20)
attr Tile_Width = 8                  # Width of each facade texture tile

@Range(0, 5)
attr Roof_Floors = 1                 # Number of floors within roof volume

@Range(0, 45)
attr Roof_Angle = 30                 # Roof pitch angle in degrees

@Range(0, 1)
attr continuousBalcony = 1           # Whether continuous balcony is applied

@Range(0, 1)
attr ShowWindows = 1                 # Enable/disable window geometry
```

Figure 65. CGA Rules about Building Parameters Section

In the definition of attribute parameters, since the construction year of each building is unknown, the building "age" is randomized to introduce visual diversity. For the overall building height, it is necessary to link the attribute values contained in the building shapefile exported from ArcGIS Pro. As illustrated in Figure 66, this linkage is established within the CityEngine Inspector by connecting the height parameter of the building model to the corresponding attribute from the shapefile.

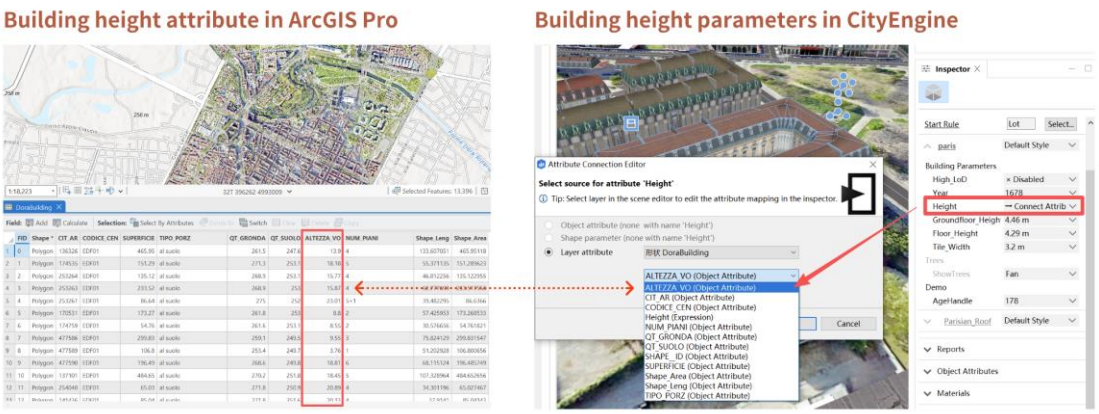


Figure 66. The height attribute of buildings in CityEngine is linked to ensure consistency with real-world conditions.

Once the total height is defined, the script must automatically calculate the distribution of floor levels. As illustrated in Figure 67, the logic works as follows:

1. Groundfloor_Height

A random value between 4.4 m and 4.7 m is assigned, reflecting the typically higher ground floors found in Turin's historic buildings

2. Floor_Height

The remaining height is obtained by subtracting the Groundfloor_Height from the total building height. This value is then divided by an average floor height of 4 m, and the output is rounded to give the number of upper floors.

```
@Range(min=4, max=6, restricted=false) @Distance
attr Groundfloor_Height = rand(4.4,4.7)   # ground floors are typically ~4.5 higher

@Range(min=3, max=5, restricted=false) @Distance
attr Floor_Height = _getInitialFloorHeight

_getInitialFloorHeight =
  (Height - Groundfloor_Height) / rint((Height - Groundfloor_Height) / 4)
```

Figure 67. CGA Rules about building height

Finally, all buildings are batch-generated as volumetric forms with realistic heights. If a high LOD model is required, it can be enabled in the parameter settings. The system will automatically recognize and match the corresponding local .obj files to the roof components, thereby enhancing the architectural detail of the generated mode for specific display (Figure 68). Among them, newly constructed modern buildings with flat roofs are manually adjusted in the Inspector panel to modify the roof type.

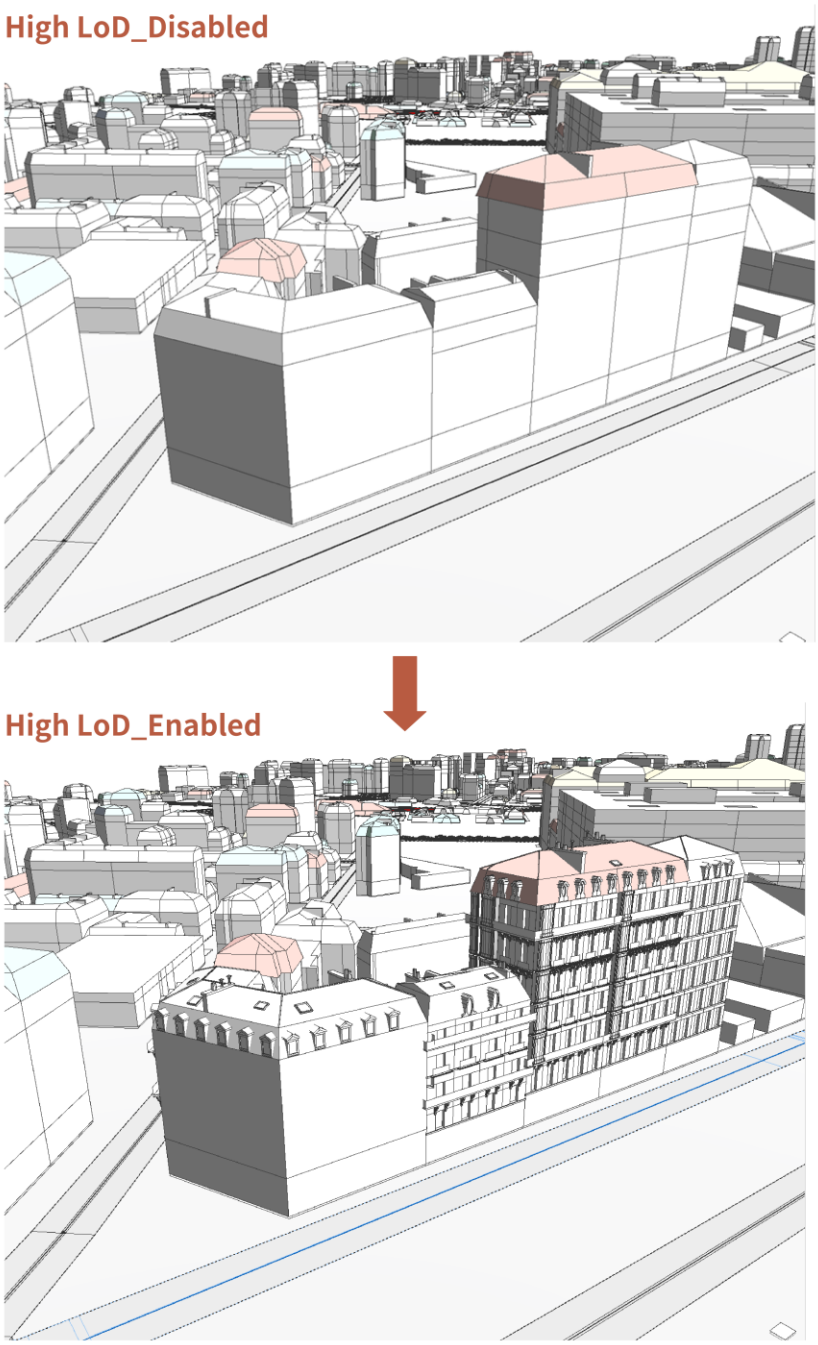


Figure 68. The High LoD models with actual heights include both the building body and the roof.

3. Defining the texture mapping objects to be included, such as roofs and façades, with façades referring to the different forms of the building's front, rear, and side elevations; then determine the representation of the roof and façades. The treatment of building façades is as follows: based on photographs of real building exteriors or similar style facades, floors and detailed components are segmented and combined, rules are automatically generated, and fixed as well as variable parts are determined to form rule templates. For certain areas of the model, externally imported components can be added.

Like the workflow used for streets, the construction of building façades also relies on breaking elements into smaller units and assembling them in a structured way. Components such as windows, balconies, and wall panels are first separated into discrete pieces and arranged horizontally and vertically to make up a single floor. Once the floor modules are organized along the X-axis, they are stacked along the Y-axis to form the complete façade.

In essence, facades generation relies on this sequential process of " split-recombination." Therefore, to build a reliable rule library, it is necessary to collect representative facade samples first, then split them into operable components, and through recombination, make the facades of different orientations of buildings integrate into a unified and coherent architectural expression.

In facade texturing, selecting facade textures based on the building's construction year and number of floors, then projects them onto the vertical surfaces.
The code shown in Figure 69 illustrates the logic used to assign materials to the building volumes.:
1. To obtain façade images locally: as previously described, the local assets folder contains façade images classified by decade (e.g., 160_1.jpg, 180_2.jpg). According to the construction year, a corresponding texture from the assets/FacadePool/ directory is applied to the façade surface.
2. In applying textures to the buildings, the *setupProjection* and *projectUV* operations are used to generate the UV coordinates needed for image mapping. These commands align the façade textures with the geometry by taking into account the actual building width and the vertical subdivision of each floor, so that the images fit proportionally onto the modeled elevations.

```
gettexture =
  case Year >= 1940: "FacadePool/" + texID + ".jpg"
  case comp.sel == "back": getCourInterieureTexture
  case Year >= 1600: "FacadePool/" + texID + ".jpg"
  else: "1_185_1.png"

FacadeSets(textureOffset) -->
  texture(gettexture)                    // calls assets/FacadePool/
  setupProjection(0, scope.xy, textureWidth, 'floorsPerTexture/nFloors, 0, '-textureOffset/nFloors)
  projectUV(0)
  Floor(textureOffset)
```

Figure 69. Rules for façade texturing

In reality, not all buildings possess richly articulated façades; some feature blind walls instead. Blind wall is a solid outer wall lacking openings like windows or doors like Figure 70, thus ensuring material continuity while minimizing unnecessary detail. The process of assigning textures follows the same logic as described above (Figure 71), except that the images extracted from the folder differ accordingly.

Figure 70. An example of a blind wall

```
getGenMurAveugleTexture = "genMurAveugle" + (ceil(rand(1,5))) + ".png"

BlindFacadeTextured -->
  alignScopeToAxes(y)
  setupProjection(0, scope.xy, 16, scope.sy, 0, 0, 16)
  projectUV(0)
  texture(getGenMurAveugleTexture)  // uses assets/genMurAveugle*.png
```

Figure 71. Rules for façade texturing

About Roof Texturing, it is necessary to apply realistic roof materials such as tiles, zinc, or slate, and to adjust the color tone to introduce visual variation. The code shown in Figure 72 illustrates the roof texturing process, where the assignment of materials is randomized.parisianRoofs folder includes multiple roof textures，such as zinc, tile.

```
// From Parisian_Roof.cga
const getRoofTex = fileRandom("assets/parisianRoofs/" + Covering + "*.png")

RoofCovering -->
  texture(getRoofTex)            // calls assets/parisianRoofs/
  color(chosenColorTone)
  setupProjection(0, scope.xy, 16, textureHeight)
  projectUV(0)
  texture(getGenMurAveugleTexture)  // uses assets/genMurAveugle*.png
```

Figure 72. Rules of assigning materials to roof

However, in reality, the residential buildings in the Dora area typically feature red-tiled roofs, a characteristic that also aligns with the majority of residential structures throughout the city of Turin. Therefore, it is necessary to modify the code so that the material extraction process

exclusively utilizes red roof tiles (Figure 73). For the small number of buildings with other roof styles, individual adjustments can then be made separately.

```
// Fixed to use red tile texture only
const getRoofTex = "assets/parisianRoofs/tile.png"

RoofCovering -->
  texture(getRoofTex)
  color(chosenColorTone)
  setupProjection(0, scope.xy, 16, textureHeight)
  projectUV(0)
  Covering.
```

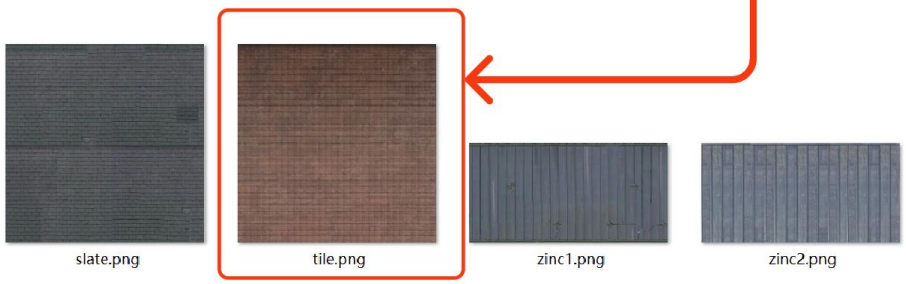


Figure 73 Rules for roof texturing

In this modified rule, the roof texture is fixed to a single red-tile image rather than randomly selected from multiple textures. By defining a constant path to tile.png under the assets/parisianRoofs/ directory, the roof appearance becomes consistent across all generated buildings, which is particularly useful for emphasizing stylistic coherence in urban-scale visualizations（Figure 74）.

Figure74. Comparison of buildings generated by CityEngine and Google 3D Map

For other materials not available in the existing asset library—such as the green titanium-zinc alloy roof of the residential buildings south of Environment Park (Figure 75)—a distinct approach is

required. Since this roof type differs entirely from the red-tiled roofs in other areas, it is necessary to import the specific material manually and create a new roof rule file to apply the appropriate texture and properties to these buildings.

Figure 75. The green titanium-zinc alloy roof visible in Google Earth.

For collecting façade materials, high-resolution orthoimages can be used to capture portions of roofs or walls. Which are imported into the project's asset library and renamed in a consistent manner, making it easier to reference them within the parametric rules and apply them to the building models (Figure 76).

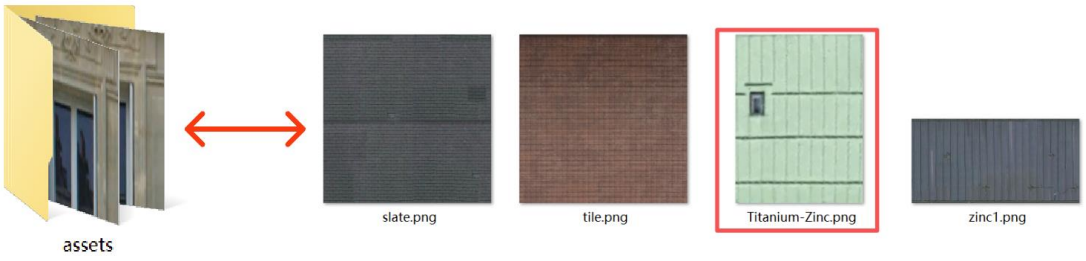


Figure 76. New material for roof texturing

Writing New Rules: a new rule file must be created (Figure 77).

1. The new roof rule file is integrated into the existing architectural rule set and saved under the name green_titanium_zinc_roof.cga. To ensure that the rule can be reused and modified in future stages, several common roof geometries—such as gambrel, mansard, terrace, gable, and hip—are included from the outset so that adjustments can later be made directly through the attribute panel.
2. The material constant refers to the texture file Titanium-Zinc.png, which is taken from the project's asset library and defined as the roof covering for the rule.
3. Functions are introduced to control the projection scale of the texture in both the X and Y directions. Because smaller values produce denser patterns, the scale factor is set to 1 to obtain a clear and visually balanced material appearance.
4. Four adjustable roof attributes—type, covering, angle, and overhang—are defined, each with an appropriate value range (for example, an overhang that can be modified between 0 and 0.5 m).

These parameters enable the roof geometry and material representation to be further refined and adjusted during the parametric modeling process.

```
###############
# related to architecture
import Green_Roof: "green_titanium_zinc_roof.cga"
  ( Type = case Year < 1960: 60%: "gambrel" 30%: "mansard" 7%: "terrace" 2%: "gable" else: "hip" else: "terrace"
  , Level_of_Detail = case High_LoD:"high" else: "low" )


###############
# constants
const getTerraceTex   = "flatroof" + ceil(rand(5)) + ".png"
const getRoofTex   = fileRandom("assets/Roofs/Titanium-Zinc.png")


###############
# functions
textureWidth      = scope.sx/ceil(scope.sx/Tile_Width)*1   # calculates the number of Bays (or tiles ) in a facade
textureHeight     =
  case Covering=="Titanium-Zinc": scope.sy*1
  else: scope.sy*6
mansardAngle      = case Angle*3<65: 65 case Angle*3>80: 80 else: Angle*3
highLOD           = case Level_of_Detail=="low": false else: true


###############
# Attributes
@Enum("hip", "gable", "mansard", "gambrel", "terrace") @Order(2)
attr Type           = 60%: "gambrel" 30%: "mansard" 7%: "terrace" 2%: "gable" else: "hip"

@Enum("zinc", "slate", "tile", "Titanium-Zinc") @Order(3)
attr Covering       = 20%: "zinc" 20%: "slate" else: "Titanium-Zinc"

@Range(min=15, max=35, restricted=false) @Angle @Order(4)
attr Angle          = case Type == "gambrel": 15 else: rand(20,35)

@Range(min=0, max=0.5, restricted=false) @Distance
attr Overhang       = case Type=="mansard" && p(0.3): rand(0.1,0.3) else: 0
```

Figure 77. Rules for new material roof texturing

It can be find from the following Figure 78 that the material of green roof has been modified successfully, and its appearance is close to reality.

Figure 78. Display of the modified new material roof in CityEngine

- 4.3.6 Landmark buildings importing

About complex landmark buildings, it is not the advantage of CityEngine parametric modeling, so you need to import existing models from SketchUP. A landmark building is often defined as a structure that stands out due to its unique design, high visibility, or historical/cultural significance[42]. From the perspective of urban semiotics, landmarks serve as distinctive spatial signs or reference points that aid orientation in both familiar and unfamiliar environments [43].
In addition, the visual impact assessment method emphasizes that the visibility of landmark buildings has a significant impact on the overall characteristics of the urban landscape and the way observers understand the urban form. However, such buildings often have irregular volumes and complex elevation details that make them difficult to accurately generate through code-based parametric modeling. Due to the lack of reusable modular components, rule-based generation is inefficient in dealing with such buildings and difficult to bring substantial benefits.
For the above reasons, it is more suitable to use professional modeling software such as SketchUp or Rhino to construct landmark buildings and other building forms with complex geometric structures. On these platforms, geometry and material mapping can be more finely controlled. The completed model can be exported to. obj and other formats, and then imported into CityEngine, so that it can keep the same spatial characteristics and visual recognition as real buildings in the overall city model.
Following this approach, the two representative structures introduced earlier in this thesis—Curia Arcivescovile di Torino and Environment Park—can be sourced from the online 3D Warehouse library. After importing these base models into SketchUp, they can be refined and corrected before their final integration into the CityEngine scene (Figure 79).

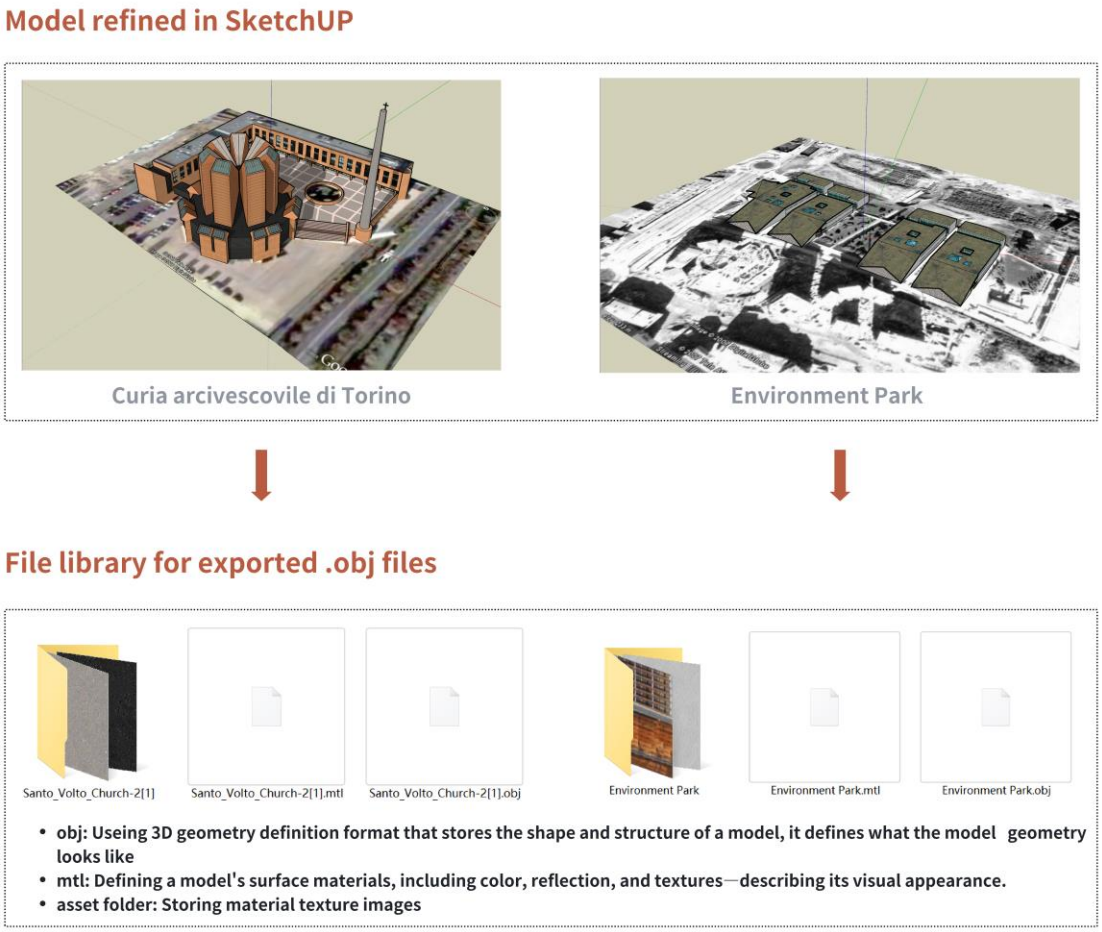


Figure 79. The process of Importing detail model files into CityEngine

To sum up, through the establishment of terrain, the adjustment of road to make it conform to the realistic section width, and texturing of building facade and roof materials, the parametric model of the whole case site is generated. Figure 80 below shows an axonometric view of the entire area, as well as individual local scenes. By completing the parameterization setting (CGA settings

for different projects) of specific areas, nearly 2,600 buildings were generated in less than a minute. And also the existing parameterization can be applied to the planning of new buildings and roads in cities in the future efficiently , and even applied to other similar cities.

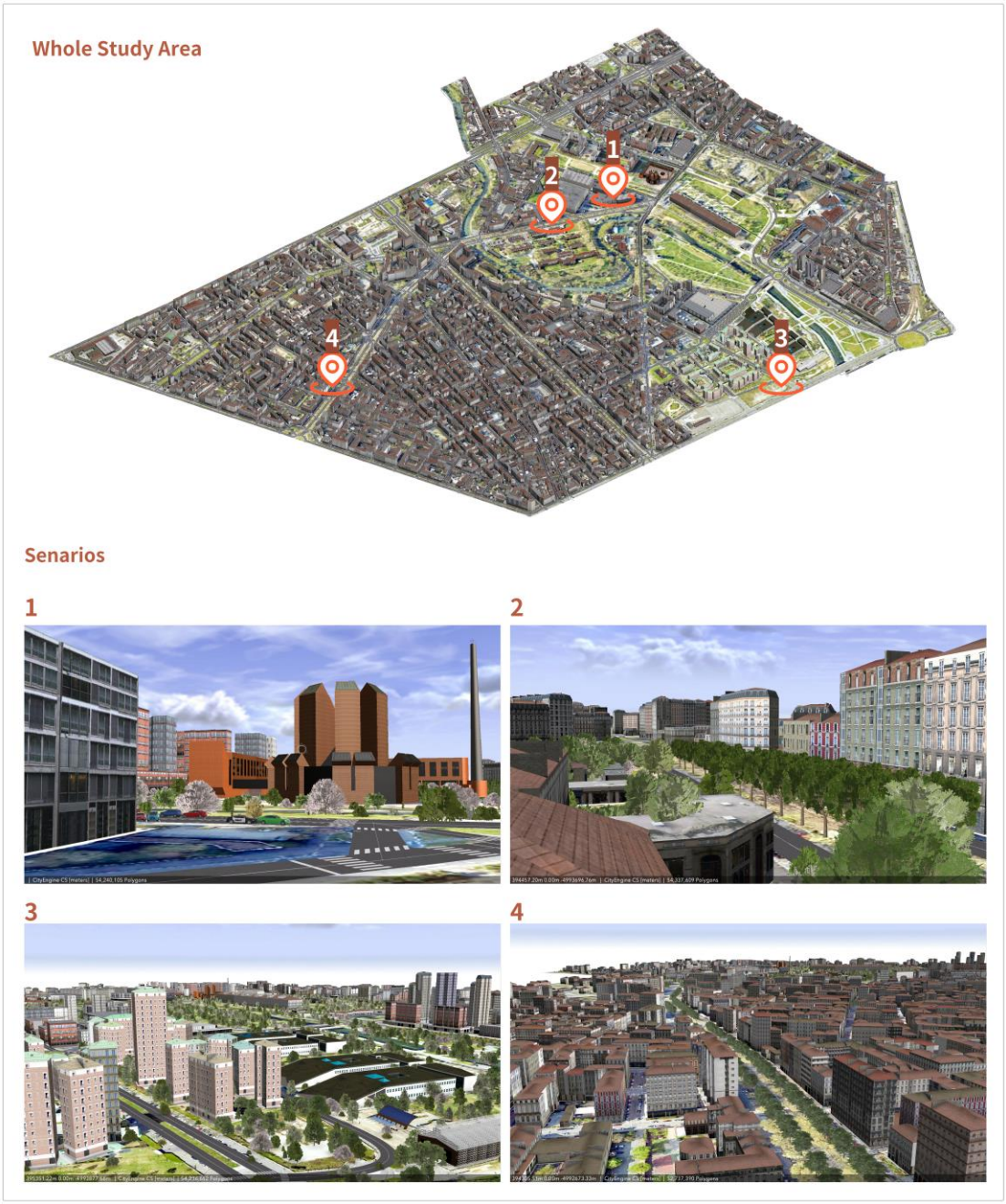


Figure 80. The Results of Parametric Modeling in City Engine

- 4.4 Three-dimensional analysis in CityEngine

- 4.4.1 Measure in 3D

3D Measure allows precise calculation of distances, heights, surface areas, and volumes within the 3D environment. It is used to verify compliance with urban design regulations, such as building heights or road widths.

Enter Analysis → Measurement, the distance tools and area measurement tools in a 3D environment can be used. (Figure 81).

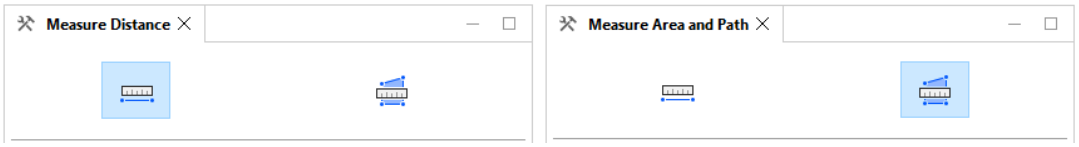


Figure 81. Measure Distance tool options

There are a few measurement types included in these tools.

1. Distance (between two points)

Diagonal distance	Diagonal distance between the points.
Horizontal distance	Horizontal distance between the points.
Vertical distance	Vertical distance (height difference) between the points.
Show laserlines	Turn laser lines on and off. The laser lines project the current height of the mouse onto the surrounding geometry.

As shown in Figure 82, the distance measurement tool was used to measure the distance between the church's sculpture and the nearest clerestory window, immediately obtaining dimensional data across multiple axes: Diagonal distance=46.41m, Horizontal distance=42m, Vertical distance=19.37m.

Figure 82. Using the Measure Distance tool to get the distance from elements

2. Area (of surfaces or polygons)

Perimeter	Perimeter length of the polygon.
Area (projected)	Area of polygon.
Show laserlines	Turn laser lines on and off. The laser lines project the current height of the mouse onto the surrounding geometry.

In Figure 83, At Environment Park, one of the key buildings is partially buried and shaped by an irregular polygonal layout formed through several straight three-dimensional segments. Because of this geometry, obtaining the perimeter and the projected extent of the green roof is not straightforward with standard measurement tools. In CityEngine, however, the task becomes manageable: by tracing the outline of the vegetated roof surface with the area measurement tool, the software directly returns the relevant values. For this building, the measured perimeter is 408.91 m and the projected area is 6,124.74 m^2.

It is important to focus that the result precision depends on the nominal map scale of the underlying data, quality of GIS data and the LOD of models. In addition, since most of the generated buildings use a uniform set of rules, the accuracy and precision do not completely reproduce the real scene. In this case, measuring small-scale dimensions may result in larger errors. Therefore, it can be used as a data reference at the block or city scale as much as possible.

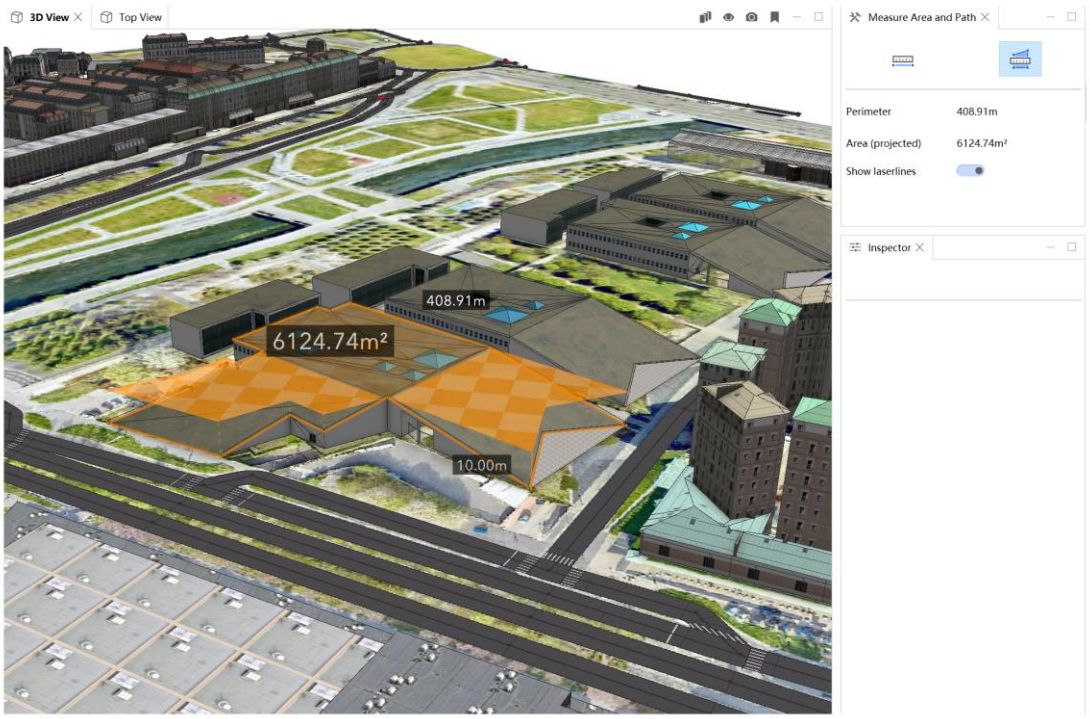


Figure 83. Using the Measure Area and Path tool to get the Three-dimensional size of the earth-covered structure.

This approach still has some limitations. The reliability of the measurement results also depends to a large extent on the quality of the DEM: even small elevation errors can be magnified in the final results. In addition, a uniform unit of measurement must be used for all spatial data, otherwise the calculation results will not be consistent. When more accurate mass or surface area evaluation is required, it is usually necessary to export geometric data to ArcGIS Pro or professional CAD

software and use more advanced analysis tools for verification.

- 4.4.2 Dashboards

The Dashboard in CityEngine is an interactive visualization panel that presents charts and key performance indicators (KPIs) based on CGA report variables. Through these panels, designers or planners can view the update results in real time when the model or attributes change, and compare and evaluate different design schemes. Dashboard can effectively assist urban design decisions, such as displaying key indicators such as land area, floor area ratio (FAR), number of building floors, building height and distribution of use functions. This thesis discusses the visualization of FAR and road features in Dashboard.

Steps in CityEngine:
1. Definition of Report Variables in CGA Rules.
The first step is to embed the report statement in a CGA rule. These statements are used to specify the metrics to be extracted, such as total building height, usable floor area, or number of residential units, and enable them to be summarized for presentation in subsequent steps.
For instance, Report ("TotalFloorArea", geometry.area * floorCount). The floor area is recorded and then accumulated and summarized in the Dashboard. Because the calculation of floor area ratio (FAR) requires " total floor area / total land area", users must first obtain the total value of all floor areas. For this reason, this study developed a new CGA rule for the DoraBuilding Shapefile: generate floors from the bottom of the building based on the number of floors per building, and accumulate the results during the calculation process to facilitate the final calculation.

Attribute definitions (Figure 84) include:
-floorHeight: randomly assign the floor height between 4 and 5 meters.
-nFloor: specify the number of building floors; A random initial value is given first in the rule and then associated with the actual number of layers attribute in the Dora Building shapefile.
-vizMode: controls the visualization mode for displaying simplified massing or detailed floor structure.

```
###############
# Attributes


@Group("Building attributes")
@Distance
attr floorHeight = rand(4,5)


@Group("Building attributes", 1)
attr nFloor = ceil((Height)/(Floor_Height))


@Group("Viz",4)
@Enum("massOnly", "floors", "massAndFloors")
attr vizMode = "massAndFloors"
```

Figure 84. Rules for the definition of attributes

Generate floor models and reports (Figure 85):

-Mass: the volume is divided step-by-step along the Y-axis based on the value of floorHeight, producing a sequence of “Floor” sub-shapes that correspond to each level of the building.

-Floor: for every level, only a single base surface is created. This surface serves as the reference geometry for computing the total floor area and, subsequently, the FAR.

-FloorBottom: compiles the cumulative Gross Floor Area (GFA) and the FAR values, and outputs them in a summarized report.

```
###############
# Mass / Floors


Mass -->
  MassViz
  split(y){ floorHeight: Floor }*


Floor --> comp(f){ bottom: reverseNormals FloorBottom }


FloorBottom -->
     report("GFA",geometry.area)
      report("FAR",geometry.area/4416483)
      color("#4444ff")
      FloorViz
```

Figure 85. Rules for generating floors and reports

As shown in Figure 86, the final output displays the generated building models with the corresponding number of floors. The report indicates that the total GFA of the analyzed region reaches 3,443,967 m^2, and the overall FAR is 78%. Furthermore, within the Dashboard interface, both the FAR and GFA values are automatically computed and visualized, providing an intuitive understanding of the area’s development density.

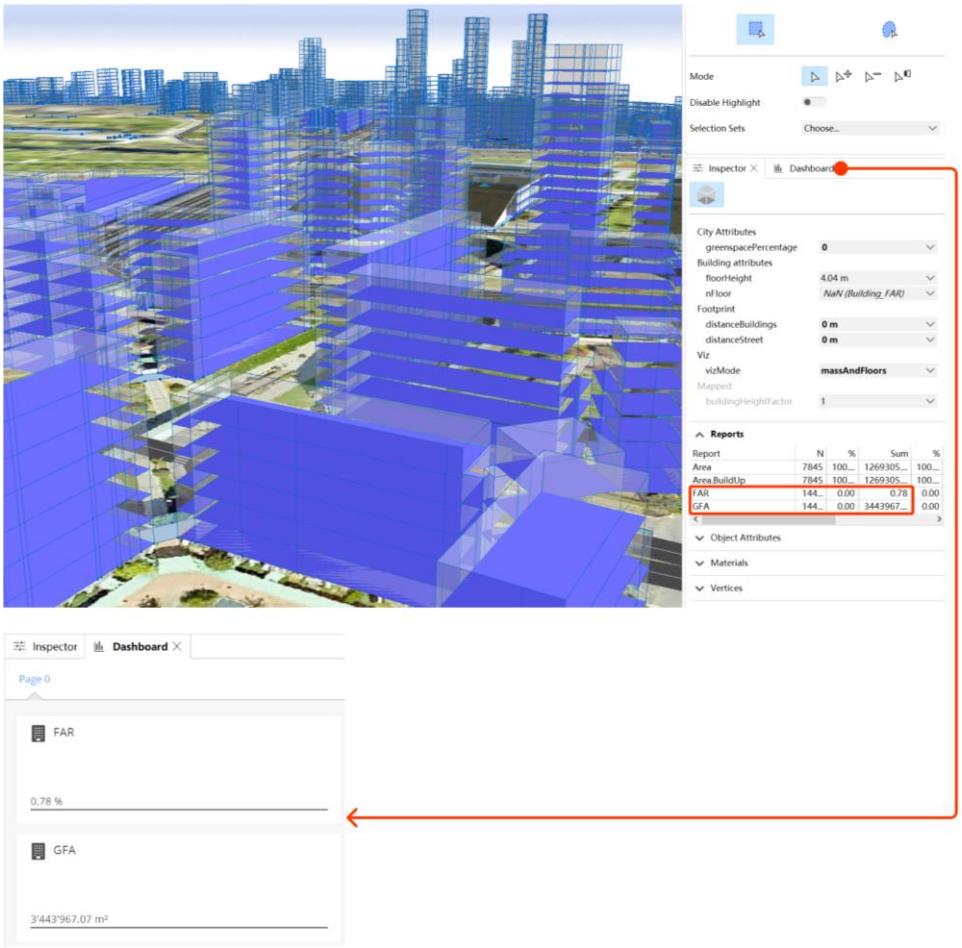


Figure 86. Site GFA, FAR reports in Dashboard

2. Configuration of Dashboard Cards

Within the dashboard interface, users create "cards" to represent specific report values. Cards can take different forms—single number indicators, bar charts, or pie charts—depending on the type of metric being represented. At this stage, the designer determines which attributes should be visualized globally across the scene (Table 5).

Icon & Title	Choose an icon and title.
Report	• Choose from a list of reports created by the applied CGA rule or rules (for example, Area by flooring type, Area by interior space type, Number of work places, and so on). • The drop-down menu has a text box that allows text filtering among the report names. • A **Key Number Card** can display the values of all reports. • The **Stack Chart**, **Bar Chart**, and **Pie Chart** only display the values of group reports. To create a group of reports, users must create reports with the syntax Group.SubReport, for example: ○ FAR. Office

	○ FAR. Resdential ○ FAR. Commerical
Divide by	• Divide a report by another report. • By using the reportName.* notation, corresponding subreports (with the same subreport name) can also be divided.
Multiply by	Multiply the values of a report by a constant number.
Reporter	You can choose to display the values of the reports computed on the entire scene or only on the selection. By default, a card displays the values on the entire scene. • **All Objects**—Default, means values from all objects in the scene are aggregated. • **Selected Objects**—Only values from selected objects are aggregated.
Aggregation	The aggregation type defines how to combine all the reported values from the whole scene into a single number for each report or subreport, that will then be shown in the chart: • Sum • Count • Mean
Unit	The **Unit** displayed next to the value in the card. Has no influence on the calculation or the value displayed.
Minimum and Maximum (for Bar Chart)	Defines the range of the value axis. • **Automatic**—Automatically adjust the value according to the displayed value.. • **Custom**—Use user-defined fixed values.
Scaling (for Bar Chart)	Specifies the scaling of the axis: • Uniform (by default) • Logarithmic
Notation	• Decimal (for example, 48'876'592) • Scientific (for example, 4.888e+7)
Round values	**Decimal places**—Specifies the number of decimals to display.
Sorting (Pie Chart)	Sorting of reports:

	• **Alphabetical**—(Report names) • **Ascending**—Lowest value first, highest last • **Descending**—Highest values first, lowest last
Legend (Stack Charts and Pie Charts)	• Display a legend in charts. • The legend adjusts automatically to the reported values.

Table 5 Parameters in Dashboards

Taking the previously generated road network as an example, the Dashboard settings allow the visualization of the proportion of different parking area types as well as the distribution ratio of various road categories within the area (Figure 87).

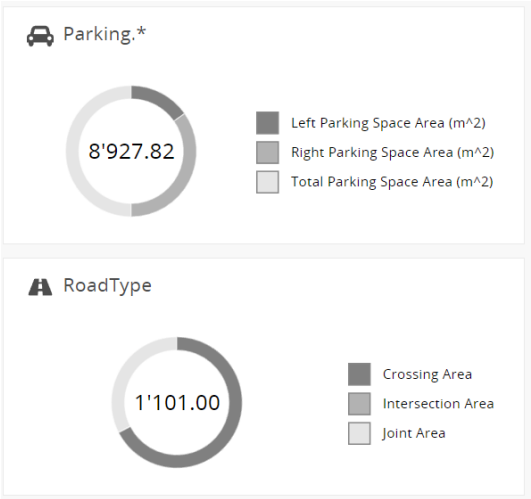


Figure 87 Part parameters of the road in the dashboard

CityEngine's Dashboard and Scenario Analysis features provide dynamic, data-driven tools for evaluating urban design alternatives. Within a single project, users can create multiple design scenarios and configure dashboards to display comparative report values across them—facilitating the assessment of how different planning strategies, such as zoning or density variations, influence key urban metrics. Moreover, the dashboards are synchronously linked to the 3D model: any modification to geometry, rule parameters, or design attributes instantly updates the corresponding analytical outputs. This real-time synchronization ensures consistency between the evolving design and its performance data, reinforcing an iterative, feedback-oriented workflow essential for comprehensive urban design evaluation.

- 4.4.3 Visibility Analysis

Visibility Analysis is used to determine which parts of the terrain or built environment can be seen from a specific observation point and which areas are out of sight. In urban studies, this type of analysis is often used to assess the possible visual impact of view galleries, view domes, or new buildings. It is an important tool for determining building height control, shaping public space and

protecting the sight of important landscapes.

Steps in CityEngine:
1. Scene preparation
– Import topo surface and building models, and ensure that elevation data and coordinate reference systems are properly defined.
– Set observation points such as roofs, viewing positions, landmarks, etc. This study mainly analyzes the church and its surrounding environment, the residential area near the park.
2. Launch the tool
– Select Analysis → Visibility Analysis from the menu and select a specific mode (Viewshed, View Dome or View Corridor) depending on users analysis needs (Figure 89).

Viewshed Creation	Calculates a viewshed that determines the visibility from a camera-like observer for a limited field of view.
View Dome Creation	Calculates a view dome that has the same functionality as the **Viewshed Creation** tool but provides a 360° field of view.
View Corridor Creation	Creates a protected view corridor where any geometry visible in the corridor is highlighted.

Figure 89 Viewshed Creation tool in CityEngine

3. Set target points and direction of observation.
As shown in Figure 90, the observation point is placed directly within the view window - specifically at the road intersection on the southeast side of the church to simulate the pedestrian's perspective when perceiving the surrounding environment at the street level. The viewing angle parameters are set to 120 ° horizontally and 60 ° vertically, and the height of observation point is set to 1.7 meters along the Y axis, so as to conform to the average horizontal height of ordinary adults.

The resulting viewable map is displayed on the left side of the image: green areas indicate where the observer can see, and red areas indicate parts obscured by buildings or terrain.
In the Inspector panel on the right, the tool also displays the three-dimensional shape of the visual cone and gives the percentage of elements such as streets, buildings, churches, sky and ground seen from this observation point.
These quantitative results can transform visual experience into measurable indicators, and provide data basis for spatial perception analysis and urban design decision-making.

Figure 90. Viewshed visibility tool (green area is visible, red area is non-visible)

A similar workflow can be applied with other visibility tools, such as the View Dome function, to study visual perception within the residential districts around Environment Park (Figure 91). This analysis reproduces the range of views available to a pedestrian moving through the neighborhood.

As shown in the figure, people standing at residential intersections can see the green roof of the park on the north side and the linear green landscape axis nearby at the same time. This indicates that the surrounding space layout has been properly designed. This visual openness avoids the sense of closure of space and enables residents to maintain a direct visual connection with the green landscape, thus creating a more comfortable and natural living environment.

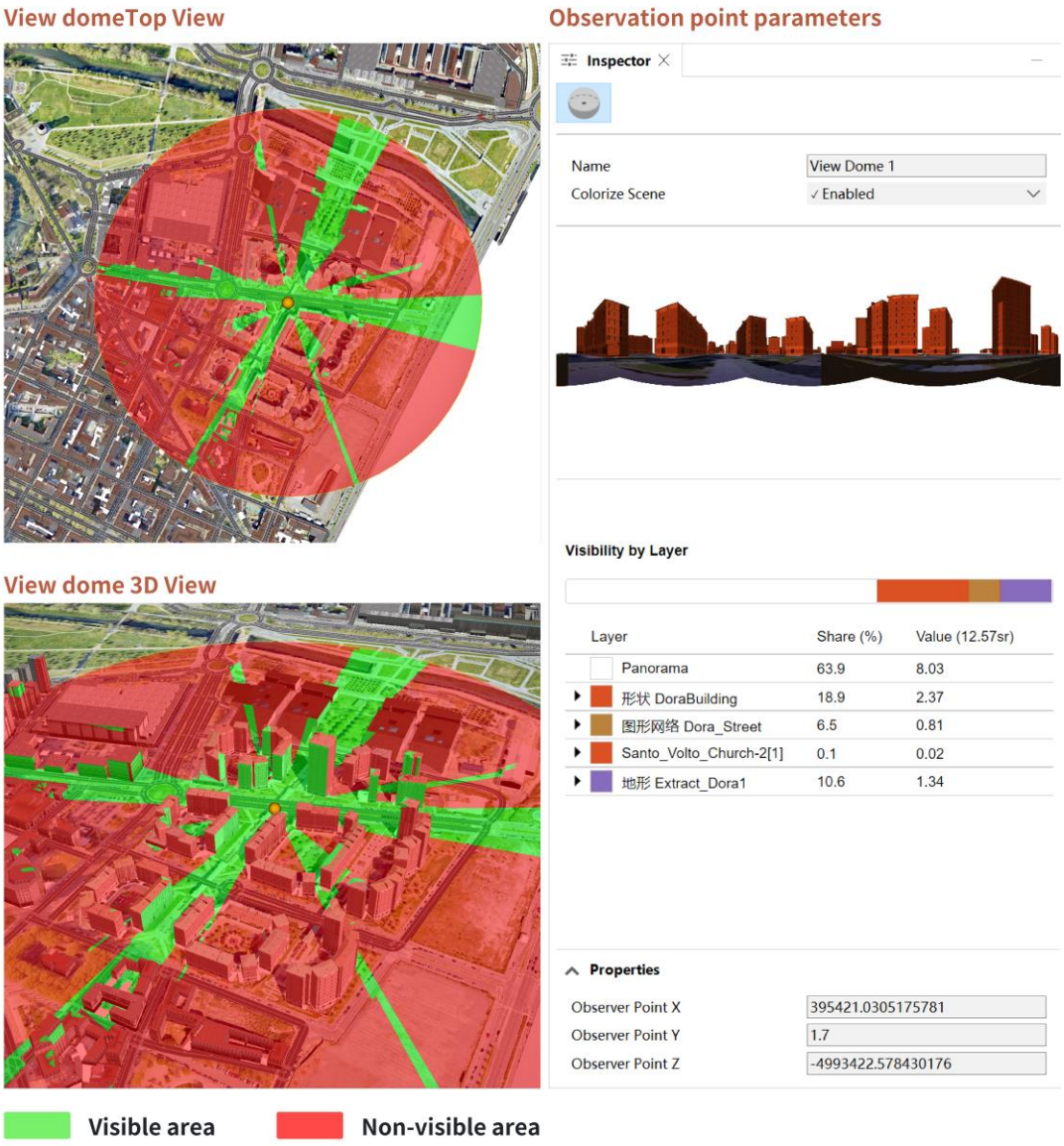


Figure 91. View dome visibility tool (green area is visible, red area is non-visible)

In this type of visual analysis, the reliability of the results is highly dependent on the quality of the underlying elevation data and the accuracy of the building geometry in the model. When the study area covers a large area of the city or includes densely built environments, the amount of computation required for analysis may increase significantly, which affects the processing efficiency or leads to a decrease in rendering speed. In addition, such outputs are based only on geometric calculations and do not simulate atmospheric conditions, light scattering or other factors that affect visual perception, so there are still some limitations in terms of visual realism.

- 4.4.4 Network Analysis

In CityEngine, the Analyze Graph function provides a quantitative assessment of the street network's topological structure through graph-based centrality metrics. Each street segment is represented as an edge in a network graph, and its relative importance within the system is computed based

on measures such as closeness and betweenness centrality. The visualization generated by CityEngine employs a color gradient ranging from red to green, where red lines indicate streets with higher centrality values—typically major urban corridors that facilitate high levels of connectivity and potential movement—while green and blue lines represent peripheral or less connected local streets (Table 6).

Color	Meaning	Concept
Red	High centrality/High traffic potential	Main roads, urban backbones, and core roads with concentrated routes
Orange/Yellow	Medium accessibility	Secondary roads or transitional roads
Green/Blue	Low accessibility	Local roads or residential side roads

Table 6. CityEngine road Analyze Graph color scheme

This form of analysis, although based on standard graph-theoretical principles, aligns conceptually with space syntax approaches that assess how street networks shape integration and accessibility within the urban system. Both methods seek to understand how the configuration of the network influences spatial hierarchy and the likely distribution of movement. By examining the resulting centrality values, it becomes possible to distinguish the streets that function as the main structural connectors of the area from those that primarily fulfill local circulation roles.

In this case, the color coded visualization results (Figure 92) show that the main north / south trunk road in the Dora region presents the highest centrality and is marked in red, indicating that it has the strongest dominant function in the urban road network structure. In contrast, smaller residential roads and interior green corridors are shown in green or cyan with low centrality values, indicating that these roads have weak traffic potential and traversal. The results of such analyses are essential for assessing urban spatial performance, helping to reveal street hierarchies, accessibility distributions and the spatial logic that underpins urban form.

Figure 92. Network analysis using Analyze Graph tool

This analysis has high requirements on raw data. Accurate network analysis in CityEngine requires clean and continuous topology, as any broken or misaligned road connections can disrupt the generation of correct spatial relationships. Moreover, CityEngine on its own offers only basic network functionalities; more advanced network metrics and spatial analyses typically need to be performed in ArcGIS Pro or other specialized GIS software.

- 4.4.5 Flood Analysis in ArcGIS Pro by data interoperability

The combination of CityEngine and ArcGIS Pro provides an efficient workflow for city-scale flood simulation and spatial risk assessment. In CityEngine, parametric rules can be used to generate a three-dimensional fine model of the city - including buildings, topographical surfaces and vegetation, which form the geometric basis for carrying out hydrological analysis. These 3D elements are then exported together with the terrain's. tif grid file as an Esri Scene Layer Package, which is read and processed in ArcGIS Pro.

In ArcGIS Pro, the 3D scene is combined with relevant hydrological data and digital elevation models to derive the flow path, potential ponding area and overall flooding extent. By integrating topographic morphology, elevation information and hydrodynamic parameters, the process supports both visualization and quantitative analysis of flood risk. This comprehensive approach links urban morphological modeling with environmental processes and provides a systematic framework for studying the response of urban built environment to extreme hydrological events.
This thesis uses the Dora Riparia River and its adjacent Environment Park as a demonstration area to verify this workflow. The steps include exporting the buildings modeled and mapped in CityEngine together with the terrain grid as an Esri Scene Layer Package to ensure that these data can be seamlessly imported into ArcGIS Pro for subsequent simulation and analysis.
During the preparation phase of flood simulation, it is necessary to configure and calibrate the simulation tools within ArcGIS Pro. As illustrated in Figure 93, the process is divided into four main stages:
1. Sketch a rectangle that covers the terrain study area
2. Set water volume parameters
3. Draw water areas and set water sources
4. Import building and terrain elevations as factors in the simulation

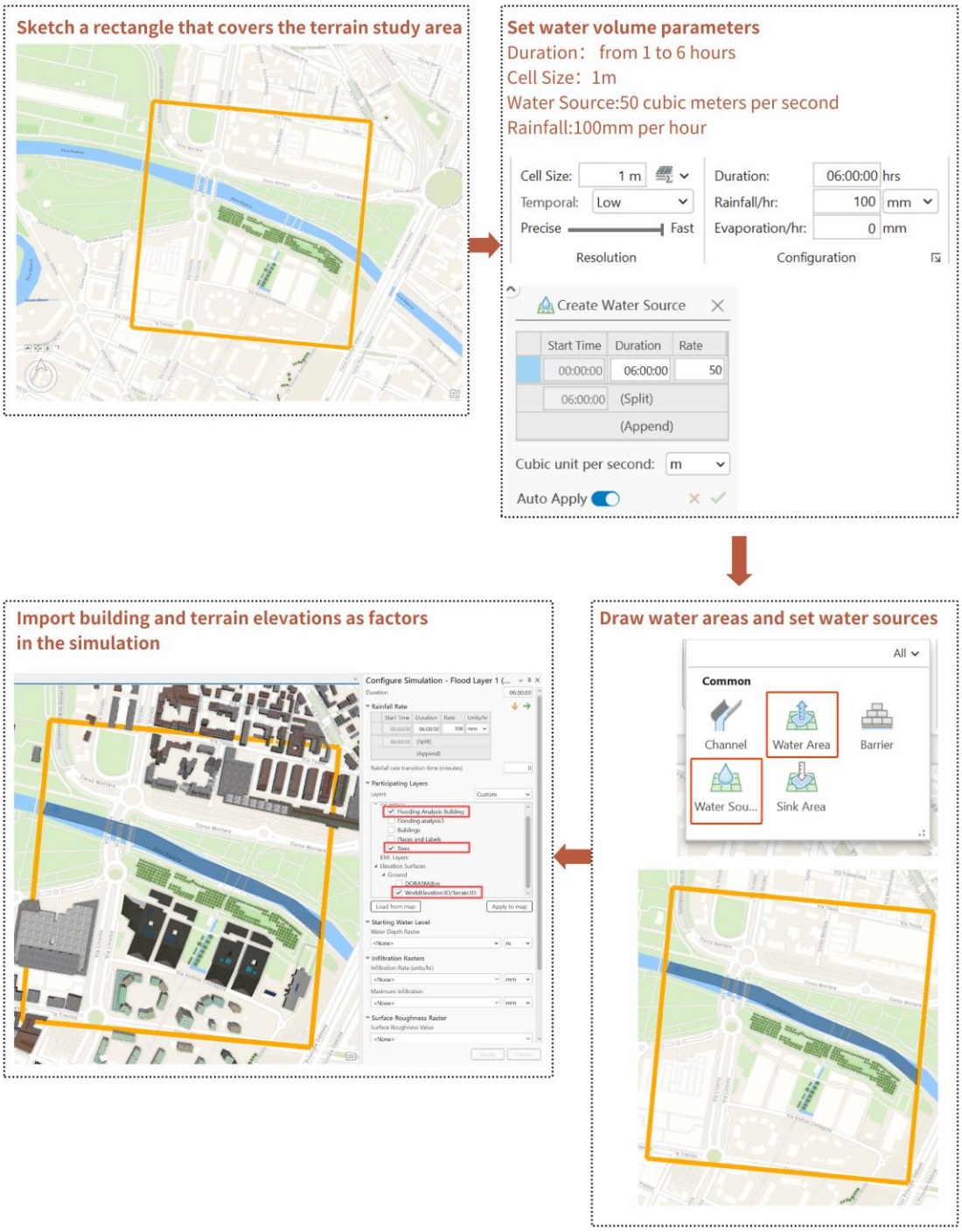


Figure 93. Flood simulation preparation phase in ArcGIS Pro

After completing all preparatory and configuration work, the simulation can be initiated. As shown in Figure 94, the water level of the Dora Riparia River gradually rises, progressively inundating the surrounding environment park areas until reaching the covered structures and nearby residential zones. The main advantage of this simulation lies in the integration of highly detailed building models generated in CityEngine, which are precisely aligned with the terrain. This allows the simulation to visually demonstrate, from a perspective view, how flooding affects buildings of different forms, façades, floor levels, and spatial enclosures. Consequently, the results provide valuable, objective, and visually interpretable insights to support urban planning decisions and

emergency response strategies.

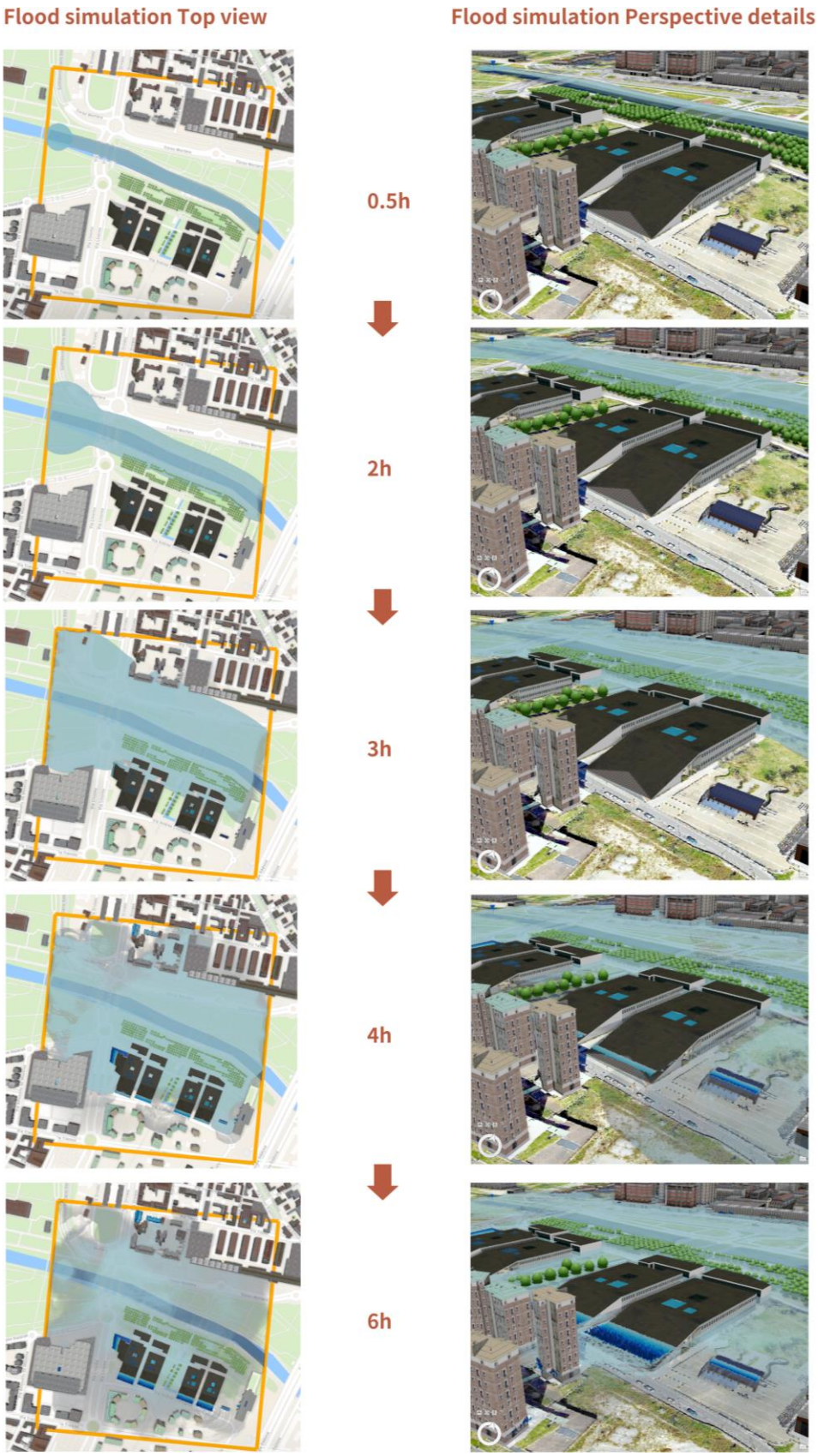


Figure 94. Flood simulation process demonstration. Top (left) and 3D (right) views

This workflow also presents several constraints. CityEngine itself does not include hydrological computation tools, meaning that processes such as water-flow modeling must be carried out in ArcGIS Pro or in dedicated hydrological software. In addition, the reliability of any terrain-based analysis depends heavily on the resolution and vertical accuracy of the DEM, when the elevation data are coarse or contain inconsistencies, the resulting flood simulations may vary considerably from real conditions. Parameters that influence water behavior—such as infiltration rates or surface roughness—usually require manual adjustment as well, since they must be calibrated to match the characteristics of the actual environment.

- 4.5 Virtual Reality asset in Unreal Engine

-4.5.1 Concept

By integrating the three-dimensional assets of the Dora region into UE5 through Cesium, an immersive and geospatial accurate virtual environment can be built, thus establishing a key connection between urban digital twins and real-time interactive visualization. CityEngine is responsible for parametric generation of city models, ArcGIS Pro provides geographical registration and spatial accuracy, while UE5 serves as the final immersive presentation layer, transforming static data into an interactive experiential environment. In this process, Cesium for UE5 acts as a central intermediary tool to transform GIS and CityEngine data into realistic 3D scenes while ensuring that geographic coordinates and terrain accuracy are preserved.

By combining high-precision geospatial data with UE5's real-time rendering technologies (such as Nanite geometric streaming rendering and Lumen global illumination), the virtual environment enables users to explore the digital twin of Dora Park in a near-real way. This immersive presentation provides a key analytical advantage: users can observe spatial relationships, visual influences, and environmental dynamics in realistic virtual scenes, such as vegetation cover or spatial scales perceived from a human perspective.

-4.5.2 Models with GIS information in virtual reality interaction

Because of the interoperability of data and coordinate systems between CityEngine and UE5, in order to achieve the purpose of virtual tour, the general process is as follows::

1. Exporting from CityEngine

The 3D city model, including buildings, streets, and river geometry, was generated in Esri CityEngine and exported as an Unreal Engine file（Figure 95）.This format preserves both geometric precision and spatial reference information, allowing seamless integration into Unreal Engine 5. Textured building assets were included to ensure consistency between GIS information and 3D environments.

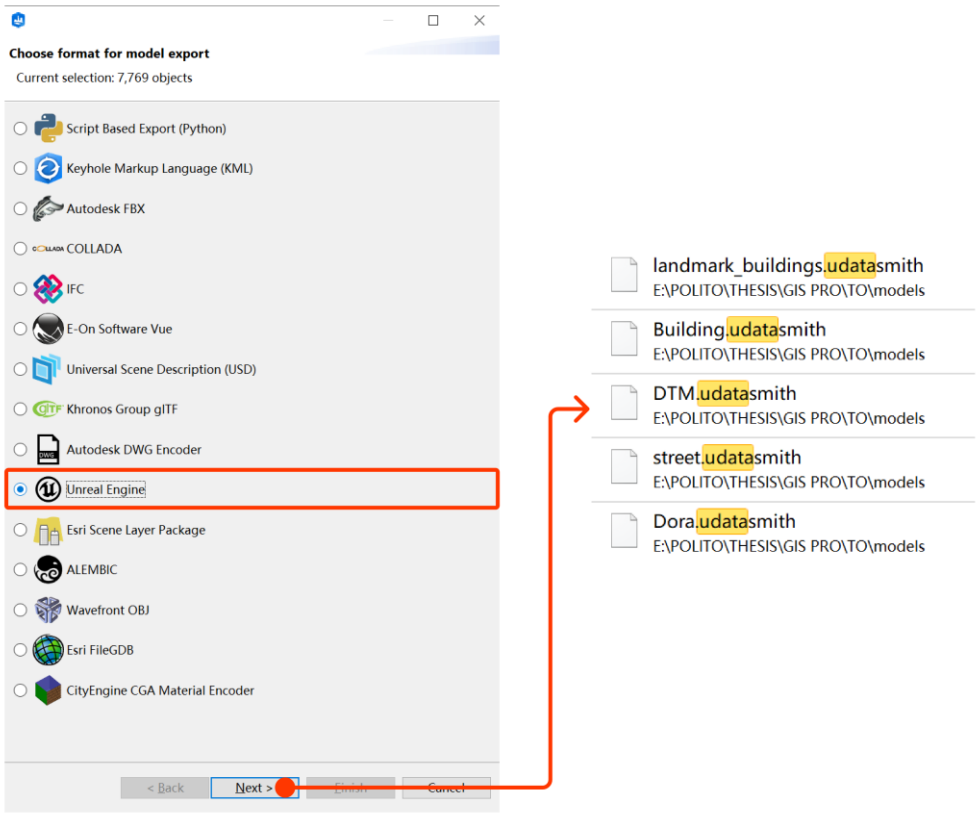


Figure 95. CityEngine exports .udatasmith files for integration with Unreal Engine

2. Importing into Unreal Engine 5

The imported assets (buildings, roads, and the Dora Riparia river) were organized into layers corresponding to their CityEngine classification. After importing, these elements will automatically align and appear in Unreal Engine (Figure 96). Textures were re-linked to maintain visual realism.

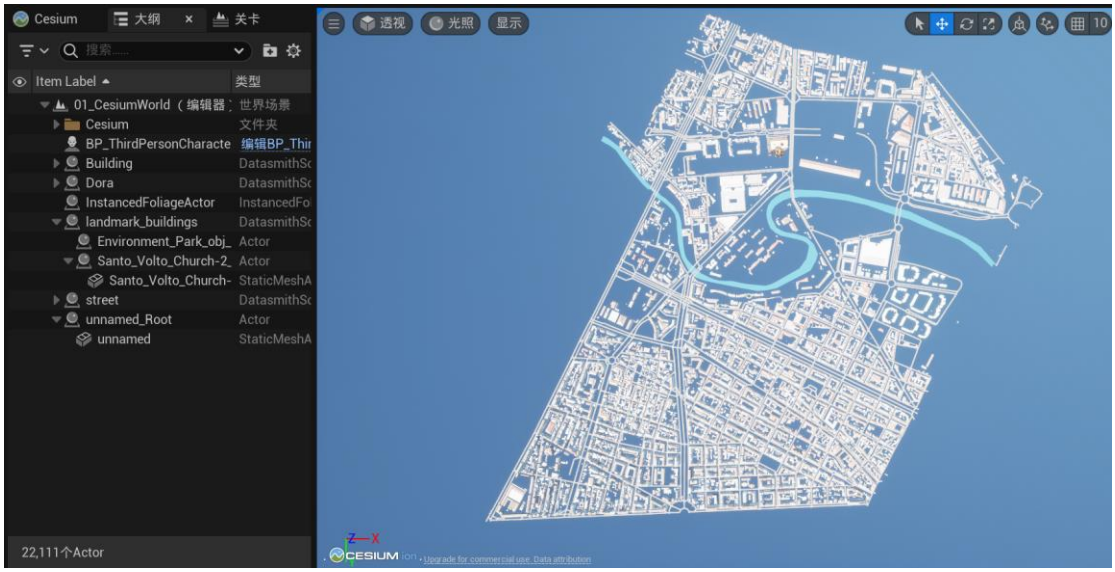


Figure 96. Importing CityEngine models into Unreal Engine

From the images above, it is clear that the imported models do not sit properly on the terrain surface and are not yet positioned within a global coordinate reference because the imported model will default to the initial world origin. At this stage, the value of Cesium for Unreal Engine becomes apparent, as the platform enables precise alignment between geographic datasets and three-dimensional models.

When generating terrain, the interface allows users to call directly from a variety of online geospatial data sources, including OSM Buildings, combined data from World Terrain and Sentinel-2 images, as well as 2D satellite imagery from Google Maps. The selection of reference layers depends on the specific research objective. Since this thesis has constructed a highly accurate building model by parametric method, only topographic surface and two-dimensional satellite images are used in the registration process. The Cesium Georeference Actor converts all coordinates imported into UE5 to the global WGS 84 geographical coordinate. By manually aligning the model's origin to the actual geographic coordinates of Turin (45.0875 N, 7.6613 E), coordinate drift and floating point accuracy error can be minimized.

This geospatial alignment ensures that the model in the virtual scene accurately corresponds to the real-world geography（Figure 97）.

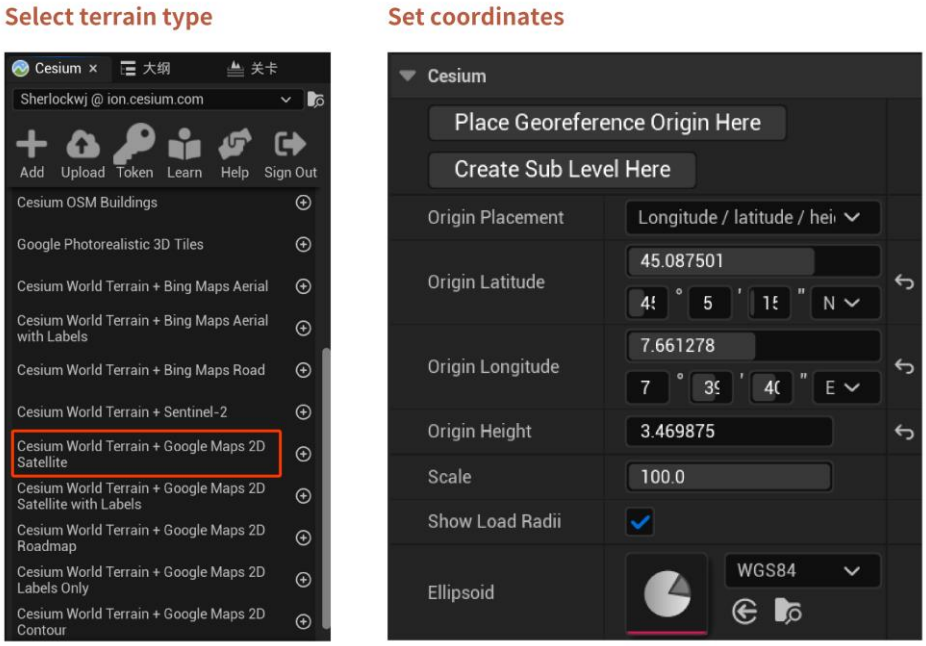


Figure 97. Setting up Turin's terrain and coordinate system in Unreal Engine

3. Terrain Alignment in Cesium

To ensure spatial accuracy, the Cesium World Terrain dataset was loaded as the base elevation model. The Globe Anchor component was added to each major asset, enabling precise placement on the terrain surface.In cases where minor vertical discrepancies occurred, the “Clamp to Terrain” function was activated to automatically project the geometry onto the real-world elevation surface. This process ensured that urban features conformed to the topographic reality of Turin’s Dora Riparia valley（Figure 98）.

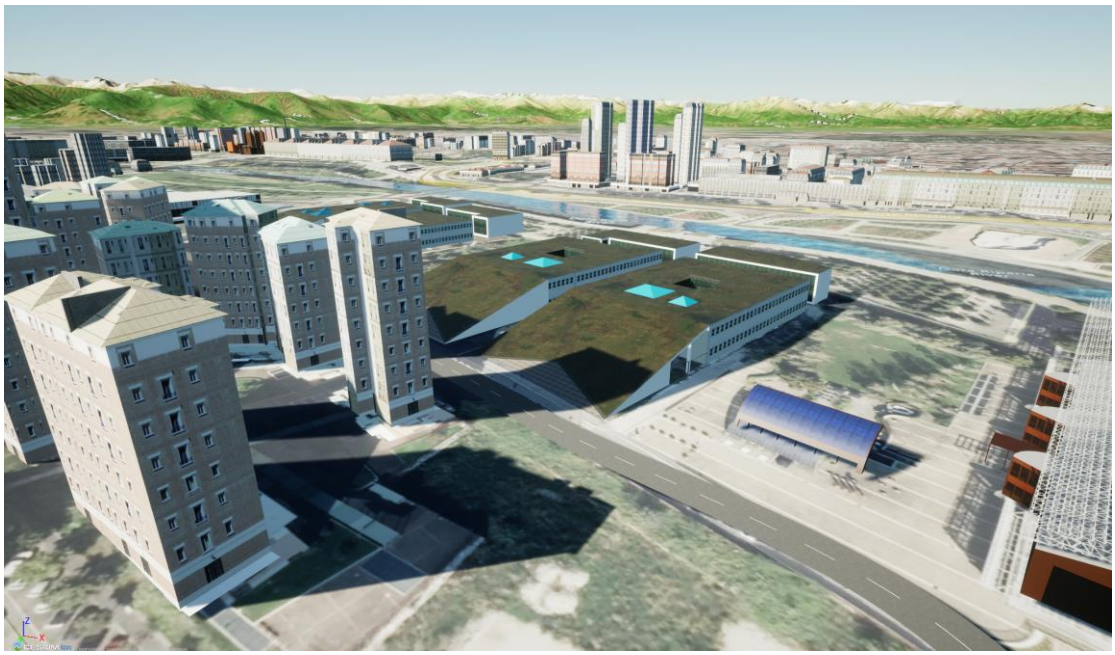


Figure 98. The model aligned to the terrain

4. Interactive Third-Person Navigation

For immersive visualization, a Third-Person Character blueprint was introduced.

This actor allowed free exploration of the environment using real-time controls—movement, rotation, and camera adjustment—offering a human-scale perspective within the virtual city. The observer height was set to 1.7 m to replicate natural eye-level viewing conditions (Figure 99).

Figure 99. Importing Third Person Character

This interactive perspective enabled users to evaluate spatial relationships, visibility, and environmental design aspects in a more intuitive manner. In terms of character operation,

supporting run, jump, and pause, allowing users to truly feel the surrounding environment in the software, Figure 100 illustrates third-person visual perspectives under different scenarios, including the environments of the Environmental Park, residential area, and the church scene.

Figure 100. Third-person visual perspectives

5. Conclusion

The combined workflow of ArcGIS Pro, CityEngine, and Cesium for Unreal Engine 5 forms a comprehensive methodological chain, linking spatial data acquisition, parametric modeling, and immersive visualization. These tools do not operate in isolation—instead, they create an interconnected system: ArcGIS Pro handles precise geospatial data processing, CityEngine enables rule-based parametric modeling, and Cesium for Unreal Engine 5 extends the results into a real-time virtual environment for interactive exploration.

GIS data show that it is an important progress in the field of 3D urban modeling to incorporate CGA-based rule modeling into the process of city construction. In contemporary urban planning, the transformation and renewal of existing urban areas are inevitable, and 3D modeling provides the most direct and efficient way for visualization and evaluation of planning schemes. When 3D modeling is combined with VR technology, urban space can be reconstructed in a highly realistic way, enabling planners to repeatedly test and optimize the spatial layout until an optimal design scheme is formed.

In this workflow, rule-based CGA modeling becomes the key link between data and design. By developing parametric rules, the city model can be dynamically adjusted according to data, reconstructed in real time, and evaluated immediately through visual dashboard, which greatly reduces the design iteration time and provides support for evidence-based decision-making. This represents a methodological shift from static modelling to parametric, data-driven design patterns where each step of modelling directly responds to quantifiable urban parameters.

At the practical level, successful implementation depends on two core factors: one is the accuracy of basic geospatial data (roads, buildings, public spaces, etc.), and the other is the ability to translate these data into parametric logic that can reflect the morphology and social context of the study area. Therefore, data acquisition, rule calibration and model generation constitute a continuous and interdependent process: the original data lays the foundation for rule modeling, and the modeling output can be further extended to immersive visualization environments such as UE5. In this sense, data processing, parametric modeling and virtual simulation should not be seen as separate steps, but rather as a unified sequence of methods in the digital twin development framework.

CityEngine has significant advantages in this process. Most of the work is concentrated in the data preparation and rule definition phase, while the final 3D model generation is relatively fast, in this thesis, defining and adjusting the rule sets required approximately two weeks. However, once this preparatory work is completed, generating similar buildings by applying the rules can be accomplished in under one minute. Moreover, when a parameter is modified, the corresponding attribute changes across all models can likewise be visualized within roughly the same short timeframe, this method significantly improves work efficiency compared to manually modifying project details at the city scale. At the same time, the reusability of CGA rules greatly improves efficiency, and existing scripts can be used as templates for subsequent projects. CityEngine has good compatibility with ArcGIS Pro, SketchUp and Unreal Engine 5 to transfer spatial data and mapping models between platforms. Although CityEngine lacks advanced analysis tools and built-in rendering functions, its parametric logic can still support visualization-oriented spatial analysis. This makes urban modeling not only a visual display, but also an analytical method.

However, this process also has some limitations. CityEngine is mainly used to generate the 3D

model of a city, which depends on the structure of CGA rules. For complex or irregular building forms, SketchUp or Rhino is often needed for supplementary modeling. At the same time, photo-level rendering and still image output also rely on external platforms such as Cesium for Unreal Engine. In addition, the accuracy of the model is limited by the quality of input data and the user's ability to process rule logic.

Overall, the multi-platform integration method provides a robust and scalable methodological framework for architecture and urban digital twins. By connecting GIS data acquisition, rule-based parametric modeling and immersive visualization, the process demonstrates how the technology path can evolve into a coherent methodology. This continuity from data to model to virtual presentation reflects the direction of digital urbanism: spatial information, parametric logic and experiential visualization converge in the same framework to support sustainable and data-driven urban design.

6. Other possible applications

The combination of CityEngine, ArcGIS Pro and UE5 provides a robust and scalable technology foundation for building urban digital twins and enabling immersive visualization. In addition to its current application in flood simulation and urban morphology analysis, this workflow also shows great potential in future interdisciplinary research.

Smart City Planning and Urban Simulation:
Future applications can be further extended to smart city management and real-time urban monitoring. By combining CityEngine's parametric modeling tools with real-time updated geospatial data, planners can simulate urban operations such as infrastructure growth, traffic flow and energy distribution under different policy scenarios. At the same time, with the real-time rendering capability of Cesium for Unreal Engine, dynamic simulation and visualization can be realized on a city scale. This approach promotes public participation in decision-making processes and supports the assessment of urban resilience.

Environment and Disaster Response System:
The framework can also be extended to broader environmental and hydrological simulations, such as air quality monitoring, landslide prediction or heat island effect analysis. When combined with IoT sensor data and predictive models, the Unreal Engine's visualization environment is capable of generating interactive risk assessment dashboards that enable emergency planners to analyze disaster propagation processes across different time scales and regions.

Cultural Heritage and Virtual Reconstruction:
The strong compatibility of CityEngine with SketchUp, Rhino and GIS formats makes it equally suitable for digital cultural heritage protection. The parametric reconstruction of historic districts (such as central Turin) and their presentation in the Unreal Engine can be used for purposes such as education and tourism. Through VR or AR, the public can explore the reconstructed historical scene in the immersive virtual space and provide a new way for cultural heritage dissemination.

Virtual Collaboration and Digital Twin Platform:
As the digital twin ecosystem continues to evolve, this workflow has the potential to evolve into a platform that supports multi-party collaboration, connecting architects, urban planners and policy makers. Through cloud integration, multiple types of participants can adjust parameters (such as building type, environmental policy or traffic mode) in real time and immediately view the results in an immersive 3D environment.

Educational Communication and Public Communication:
UE5's interactivity and high-quality visuals create new possibilities for educational outreach and community engagement. By transforming abstract urban data into intuitive visual experience, the understanding gap between technical experts and the public can be narrowed, so that citizens can play a more active function in the process of urban decision-making.

7. Bibliography

[1] "REGOLAMENTO DIDATTICO Corso di laurea magistrale in AGRITECH ENGINEERING." Accessed: Dec. 05, 2025. [Online]. Available: https://www.polito.it/sites/default/files/2025-07/Academic%20Regulations_MA_ACC_25_26.pdf

[2] M. F. Goodchild *et al.*, "Next-generation Digital Earth," *Proc Natl Acad Sci U S A*, vol. 109, no. 28, pp. 11088–11094, July 2012, doi: 10.1073/pnas.1202383109.

[3] M. Craglia *et al.*, "Digital Earth 2020: towards the vision for the next decade," *International Journal of Digital Earth*, vol. 5, no. 1, pp. 4–21, Jan. 2012, doi: 10.1080/17538947.2011.638500.

[4] "Destination Earth." Accessed: Nov. 04, 2025. [Online]. Available: https://www.esa.int/Applications/Observing_the_Earth/Destination_Earth

[5] W. Shi, M. F. Goodchild, M. Batty, M.-P. Kwan, and A. Zhang, Eds., *Urban Informatics*. in The Urban Book Series. Singapore: Springer Singapore, 2021. doi: 10.1007/978-981-15-8983-6.

[6] S. Somanath, V. Naserentin, O. Eleftheriou, D. Sjölie, B. S. Wästberg, and A. Logg, "Towards Urban Digital Twins: A Workflow for Procedural Visualization Using Geospatial Data," *Remote Sensing*, vol. 16, no. 11, p. 1939, Jan. 2024, doi: 10.3390/rs16111939.

[7] M. Grieves, *Virtually Perfect: Driving Innovative and Lean Products through Product Lifecycle Management*. 2011.

[8] M. Singh *et al.*, "Applications of Digital Twin across Industries: A Review," *Applied Sciences*, vol. 12, p. 5727, June 2022, doi: 10.3390/app12115727.

[9] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, May 2020, doi: 10.1016/j.cirpj.2020.02.002.

[10] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," in *Transdisciplinary Perspectives on Complex Systems*, F.-J. Kahlen, S. Flumerfelt, and A. Alves, Eds., Cham: Springer International Publishing, 2017, pp. 85–113. doi: 10.1007/978-3-319-38756-7_4.

[11] "Woodbury Elements of Parametric Design | PDF," Scribd. Accessed: Sept. 09, 2025. [Online]. Available: https://www.scribd.com/document/703581321/Woodbury-Elements-of-Parametric-Design

[12] "Architecture in the Digital Age Design and Manufacturing."

[13] P. Schumacher, "Parametricism: A New Global Style for Architecture and Urban Design," *Architectural Design*, vol. 79, no. 4, pp. 14–23, July 2009, doi: 10.1002/ad.912.

[14] I. Caetano, L. Santos, and A. Leitão, "Computational design in architecture: Defining parametric, generative, and algorithmic design," *Frontiers of Architectural Research*, vol. 9, no. 2, pp. 287–300, June 2020, doi: 10.1016/j.foar.2019.12.008.

[15] "Integrating models of civil structures in digital twins: State-of-the-Art and challenges," *Journal of Infrastructure Intelligence and Resilience*, vol. 3, no. 3, p. 100100, Sept. 2024, doi: 10.1016/j.iintel.2024.100100.

[16] J. Steuer, "Defining Virtual Reality: Dimensions Determining Telepresence," *Journal of Communication*, vol. 42, no. 4, pp. 73–93, Dec. 1992, doi: 10.1111/j.1460-2466.1992.tb00812.x.

[17] O. El-Said and H. Aziz, "Virtual Tours a Means to an End: An Analysis of Virtual Tours' Role in Tourism Recovery Post COVID-19," *Journal of Travel Research*, vol. 61, no. 3, pp. 528–548, Mar. 2022, doi: 10.1177/0047287521997567.

[18] C. Ouerghemmi, M. Ertz, N. Bouslama, and U. Tandon, "The Impact of Virtual Reality (VR) Tour Experience on Tourists' Intention to Visit," *Information*, vol. 14, no. 10, p. 546, Oct. 2023, doi: 10.3390/info14100546.

[19] S. Yang and H. Kim, "Urban digital twin applications as a virtual platform of smart city," *International Journal of Sustainable Building Technology and Urban Development*, vol. 12, no. 4, pp. 363–379, Dec. 2021, doi: 10.22712/SUSB.20210030.

[20] T. Deng, K. Zhang, and Z.-J. (Max) Shen, "A systematic review of a digital twin city: A new pattern of urban governance toward smart cities," *Journal of Management Science and Engineering*, vol. 6, no. 2, pp. 125–134, June 2021, doi: 10.1016/j.jmse.2021.03.003.

[21] E. Shahat, C. T. Hyun, and C. Yeom, "City Digital Twin Potentials: A Review and Research Agenda," *Sustainability*, vol. 13, no. 6, Art. no. 6, Jan. 2021, doi: 10.3390/su13063386.

[22] O. Çalışkan, "Design thinking in urbanism: Learning from the designers," *Urban Des Int*, vol. 17, no. 4, pp. 272–296, Dec. 2012, doi: 10.1057/udi.2012.21.

[23] N. Steinø and E. Obeling, "Developing a Parametric Urban Design Tool: Hybrid Environment for Architecture," *Architecturae et Artibus*, vol. 6, no. 1, pp. 51–57, 2014, Accessed: Aug. 16, 2024. [Online]. Available: http://www.architektura.pb.edu.pl/konferencja/index.php/conference

[24] R. M. ElBatran and W. S. E. Ismaeel, "Applying a parametric design approach for optimizing daylighting and visual comfort in office buildings," *Ain Shams Engineering Journal*, vol. 12, no. 3, pp. 3275–3284, Sept. 2021, doi: 10.1016/j.asej.2021.02.014.

[25] "Desktop GIS Software | Mapping Analytics | ArcGIS Pro." Accessed: Nov. 20, 2025. [Online]. Available: https://www.esri.com/en-us/arcgis/products/arcgis-pro/overview

[26] X. Song *et al.*, "An Approach for Estimating Solar Photovoltaic Potential Based on Rooftop Retrieval from Remote Sensing Images," *Energies*, vol. 11, no. 11, p. 3172, Nov. 2018, doi: 10.3390/en11113172.

[27] "Generatore procedurale di città 3D | Progettazione di città 3D per ambienti urbani." Accessed: Nov. 20, 2025. [Online]. Available: https://www.esri.com/it-it/arcgis/products/arcgis-cityengine/overview

[28] T. Kelly, "CityEngine: An Introduction to Rule-Based Modeling," in *Urban Informatics*, W. Shi, M. F. Goodchild, M. Batty, M.-P. Kwan, and A. Zhang, Eds., Singapore: Springer, 2021, pp. 637–662. doi: 10.1007/978-981-15-8983-6_35.

[29] Y. I. H. Parish and P. Müller, "Procedural Modeling of Cities".

[30] "SketchUp：3D design software – turning design ideas into reality." Accessed: Nov. 20, 2025. [Online]. Available: https://sketchup.trimble.com/zh-cn

[31] "Cesium for Unreal," Cesium. Accessed: Nov. 20, 2025. [Online]. Available: https://cesium.com/platform/cesium-for-unreal/

[32] T. Rantanen, A. Julin, J.-P. Virtanen, H. Hyyppä, and M. T. Vaaja, "Open Geospatial Data Integration in Game Engine for Urban Digital Twin Applications," *ISPRS International Journal of Geo-Information*, vol. 12, no. 8, p. 310, Aug. 2023, doi: 10.3390/ijgi12080310.

[33] J. Shan and C. K. Toth, *Topographic laser ranging and scanning: principles and processing*, Second edition. Boca Raton, FL: CRC Press, 2018.

[34] T. G. Farr *et al.*, "The Shuttle Radar Topography Mission," *Reviews of Geophysics*, vol. 45, no. 2, June 2007, doi: 10.1029/2005RG000183.

[35] M. A. Fonstad, J. T. Dietrich, B. C. Courville, J. L. Jensen, and P. E. Carbonneau, "Topographic structure from motion: a new development in photogrammetric measurement," *Earth Surface*

Processes and Landforms, vol. 38, no. 4, pp. 421–430, 2013, doi: 10.1002/esp.3366.

[36] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, Oct. 2008, doi: 10.1109/MPRV.2008.80.

[37] "Koning&Eizenberg.theLanguageOfThePrairie.pdf." Accessed: Sept. 11, 2025. [Online]. Available: https://www.andrew.cmu.edu/course/48-747/subFrames/readings/Koning%26Eizenberg.theLanguageOfThePrairie.pdf

[38] D. Bruton and A. Radford, *Digital Design: A Critical Introduction*. A&C Black, 2013.

[39] F. Biljecki, H. Ledoux, and J. Stoter, "An improved LOD specification for 3D building models," *Computers, Environment and Urban Systems*, vol. 59, pp. 25–37, Sept. 2016, doi: 10.1016/j.compenvurbsys.2016.04.005.

[40] W. Shi, M. F. Goodchild, M. Batty, M.-P. Kwan, and A. Zhang, Eds., *Urban Informatics*. in The Urban Book Series. Singapore: Springer Singapore, 2021. doi: 10.1007/978-981-15-8983-6.

[41] D. Wasserman, *d-wasserman/Complete_Street_Rule*. (Sept. 30, 2025). Python. Accessed: Oct. 02, 2025. [Online]. Available: https://github.com/d-wasserman/Complete_Street_Rule

[42] L. Gibilaro and G. Mattarocci, "Landmark Buildings and Diversification Opportunities in the Residential Market".

[43] H. A. Bala, "Landmarks in Urban Space as Signs," *Current Urban Studies*, vol. 4, no. 4, pp. 409–429, Oct. 2016, doi: 10.4236/cus.2016.44027.