Integrating Photogrammetric 3D City Models into Virtual Reality: A Methodological Approach Using Unreal Engine

İsmail Çağrı Gençtürk [1], Bilal Erkek [2], Ekrem Ayyıldız [3]

[1] General Directorate of Land Registry and Cadastre Ankara, Türkiye - tk48663@tkgm.gov.tr
[2] General Directorate of Land Registry and Cadastre Ankara, Türkiye - berkek@tkgm.gov.tr
[3] General Directorate of Land Registry and Cadastre Ankara, Türkiye – eayyildiz@tkgm.gov.tr

Keywords: Aerial Photogrammetry, Unreal Engine, Virtual Reality (VR), 3D City Models.

Abstract

Three-dimensional (3D) city models have become indispensable in urban planning, disaster management, and various geospatial analyses. Advances in photogrammetric techniques and virtual reality (VR) platforms now enable highly detailed, immersive representations of urban environments for more effective decision-making. This paper presents a methodological framework for integrating produced photogrammetric 3D building models into Unreal Engine, with the goal of creating a VR environment. The workflow covers the capture of nadir and oblique aerial imagery, the organization of photogrammetric blocks, triangulation, and stereo compilation to derive accurate roof and facade geometries. Texture mapping is then applied using nadir and oblique photographs to ensure visual realism. Models in OBJ format are imported into Blender for scaling and preparation, and subsequently exported to FBX for compatibility with Unreal Engine. The paper also discusses incorporating georeferenced data via the Cesium for Unreal plugin. Despite a currently unresolved alignment issue that prevents proper placement of the models within Cesium's global coordinate system, the outlined workflow provides a foundation for future research. Future improvements could include using pixel streaming for group VR sessions, adding GIS features to look up detailed information, and using real-time data for live updates. This method can help create interactive city simulations, making it easier for stakeholders to plan and analyze.

1. Introduction

Recent advancements in photogrammetry, geospatial visualization, and virtual reality (VR) have significantly broadened the possibilities for creating immersive 3D representations of urban environments. Three-dimensional city models, which accurately capture architectural and spatial features, now play a crucial role in urban planning, simulation, and decision-making processes (Jochem and Goetz, 2012). These models allow stakeholders to visualize and analyze complex geospatial data in intuitive and interactive ways, thereby supporting applications ranging from disaster management to public engagement in urban development projects.

Photogrammetric techniques, in particular, have emerged as a reliable and scalable method for generating detailed 3D city models. By combining nadir and oblique aerial photographs with ground control points (GCPs), photogrammetric workfows can achieve accurate representations of roofs, facades, and terrain (Mitishita et al., 2008). When used in modern game engines like Unreal Engine, these models can be viewed and interacted with in real time, making them ideal for VR platforms. Additionally, plugins like Cesium for Unreal provide geospatial functionality, enabling the precise alignment of models within a global reference system.

However, integrating photogrammetric 3D city models into Unreal Engine, and subsequently into a VR framework, poses various practical challenges. These include proper data preparation, suitable export formats (e.g., FBX), and alignment with georeferenced base maps. The latter is particularly important when combining local 3D city models with georeferenced datasets, as Unreal Engine does not natively support coordinate projections, making precise alignment with global datasets like those from Cesium for Unreal a critical challenge. This paper outlines the production of photogrammetric 3D building models, which are used in the VR environment as assets to represent detailed 3D city models. It further describes their preparation for real-time visualization and the steps required to create a VR environment in Unreal Engine. Notably, an alignment issue arose during the import of FBX files into Unreal Engine, resulting in a mismatch when placing the models into Unreal Engine environment. As of this writing, this issue remains unresolved, and the subsequent development tasks relying on correct alignment are deferred as future work. Despite this setback, the pipeline described here provides a foundation for researchers and practitioners looking to integrate detailed 3D urban data into a game engine like Unreal Engine 5.

2. Production of 3D Building Models

This chapter outlines the methodological framework for generating 3D building models using photogrammetric techniques according to the Production of 3D City Models and Creation of 3D Cadastre Bases Project (TKGM, 2025).

2.1 Ground Control Points (GCPs) and Photogrammetric Block Formation

2.1.1 Marking and Measuring Ground Control Points: Accurate measurement of Ground Control Points (GCPs) is the foundation of photogrammetric production. These distinctive field points, identified through geodetic GNSS or classical polygon surveys, ensure the geometric reliability of subsequent processes. Precise GCP positioning directly impacts the overall accuracy of the 3D city models by serving as anchors for photogrammetric triangulation and adjustment (Parvu et al., 2024).

2.1.2 Forming Photogrammetric Blocks: Aerial photographs are organized into photogrammetric blocks, grouping images based on spatial and temporal relationships. High-resolution cameras like the UltraCam Osprey Mark 3 acquire

nadir and oblique images, enabling comprehensive 3D modeling. This systematic grouping ensures consistent orientation and facilitates precise photogrammetric triangulation (TKGM, 2025).

2.2 Photogrammetric Triangulation and Adjustment

The photogrammetric triangulation process is essential for determining the position, orientation, and rotation of aerial images. To ensure geometric consistency, block adjustment—a statistical refinement method—is applied to minimize errors. This process is conducted separately for nadir and oblique images due to their distinct roles in 3D modeling:

- Nadir Photographs: These vertical images are critical forgenerating large-scale maps, orthophotos, and accurate roof surface measurements. With a ground sampling distance (GSD) of 10 cm, nadir triangulation adheres to strict regulatory accuracy standards (e.g., Regulation on the Production of Large-Scale Maps and Map Information) to produce 1:1,000-scale maps (HKMO, 2005).

- Oblique Photographs: Captured at tilted angles, these images provide detailed views of building sides, enabling realistic texturing of fac¸ades. The geometric accuracy of oblique triangulation must remain within ±3 GSD, and the integration of ground control points (GCPs) and tie points is essential when merging nadir and oblique blocks (TKGM, 2025).

2.3 Stereo Compilation

Stereo compilation involves constructing the 3D geometry of buildings from overlapping nadir images. The process begins by setting up the stereo model, which uses nadir photographs oriented with ground control points (GCPs) and triangulation results as its foundation (Toutin, 2004). From these images, 3D building vectors are created, delineating roof outlines, terraces, and other structural details for accurate representation. The data is then organized into layers, such as roof boundaries and auxiliary structures, to standardize visualization and facilitate efficient management. Throughout the process, strict adherence to detailed production manuals ensures uniformity and consistency across projects, guaranteeing high-quality outputs.

2.4 True Orthophoto Production

True orthophotos, generated from nadir photographs, are distortionfree images essential for accurate mapping. The production process begins with the use of a Digital Surface Model (DSM), which incorporates height data from buildings, terrain, and other features to rectify the images and minimize edge distortions. To ensure geometric fidelity, building boundaries are refined using specialized edge correction filters. Additionally, adjacent orthophoto sheets are created with a 30-meter overlap, enabling seamless mosaicking and facilitating rigorous quality control to maintain the accuracy and consistency of the final product (TKGM, 2025).

2.5 DSM and DTM Production

The Digital Surface Model (DSM) captures the elevation of all features, including buildings, vegetation, and terrain, while the Digital Terrain Model (DTM) represents the bare ground by removing non-terrain elements. DTMs are derived either by directly classifying and filtering out features from the DSM or through point cloud processing. Although photogrammetric point clouds are less detailed than LiDAR data, they still support effective classification for terrain modeling. To maintain consistency, DTMs must match the spatial resolution of the DSM, which is 10 cm in this study. Rigorous quality checks are performed, particularly around building footprints and sloped areas, to ensure accuracy and reliability in the final models (TKGM, 2025).

2.6 Constructing and Texturing 3D Building Models

The process of constructing and texturing 3D building models transforms vector data into detailed, realistic representations of urban environments. This involves modeling geometric structures, applying textures, and preparing models for diverse applications. Outputs are typically delivered in widely used file formats like OBJ or CityGML, with specific supplementary files for texturing and material definitions (TKGM, 2025). Below are the key steps and considerations:

2.6.1 3D Modeling of Vector Data: The geometric structure of 3D building models is derived from polygonal and line data obtained through stereo photogrammetry.

- Roof Geometry Creation: Vector data delineating roof bound-aries and elements (e.g., terraces, chimneys) are converted into 3D polygons.

- Main and Auxiliary Structures: Distinctions are made betweenprimary structures (e.g., main building) and secondary elements (e.g., rooftop installations) to improve semantic organization.

2.6.2 Texturing: Textures are applied to 3D models to enhance visual realism. Two distinct processes address the different surfaces of the buildings:

- Roof Texturing:
  - Data Source: Roof textures are extracted from nadir aerial photographs or true orthophotos.
  - Proper alignment of textures ensures that the roof’s geometric properties are preserved.

- Building Side Texturing:
  - Data Source: Oblique aerial photographs provide the imagery for realistic building side textures.
  - Resolution: High-resolution images are mapped to the side surfaces to capture architectural details like windows, doors, and exterior designs.

Textures are organized and managed using material libraries. In the OBJ format, texture-related data is stored in MTL (material library) files, which define properties such as texture images (JPG files) and their mapping to the 3D geometry (TKGM, 2025).

2.6.3 Exporting to OBJ Format: The OBJ file format is widely used for storing 3D model data due to its compatibility with various modeling and visualization platforms (Dyaksa et al., 2023). An OBJ file typically consists of:

a. Geometry Data:

- Vertex positions (v): Define the 3D coordinates of points in space

- Resolution: Texture coordinates (vt): Map textures to thegeometry.architectural
- Faces (f): Represent polygons (typically triangles or quad-rilaterals) using vertex indices.

Figure 1 represents a 3D building in .obj format without attached .mtl and texture files.

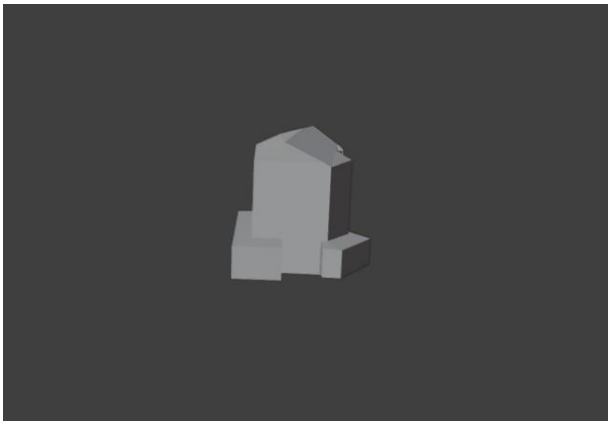

Figure 1. 3D Building Model in .obj format.

b. Material Library (MTL):

The MTL file contains material definitions used by the OBJ file. It links specific textures (e.g., JPG files) to parts of the model.

Example:

mtllib F-3017499-A.mtl

c. Texture Files:

Textures are stored as JPG or other image files. These images correspond to roof or building side textures and are mapped to the geometry using the material library.

Figure 2 represents a 3D building in .obj format with attached .mtl and texture files.

2.6.4 Preparing for Export and Quality Assurance: Export formats for 3D building models included CityGML and OBJ, each serving distinct purposes. CityGML provides a robust framework for integrating both geometric and semantic data, enabling detailed categorization of building components such as walls, roofs, and openings. This format supports structured organization and interoperability, making it ideal for projects requiring semantic richness and integration with geospatial systems (Saran et al., 2015). OBJ, on the other hand, is a versatile format widely used for visualization and rendering, capturing geometry and texture mapping effectively (Toledo et al., 2008).

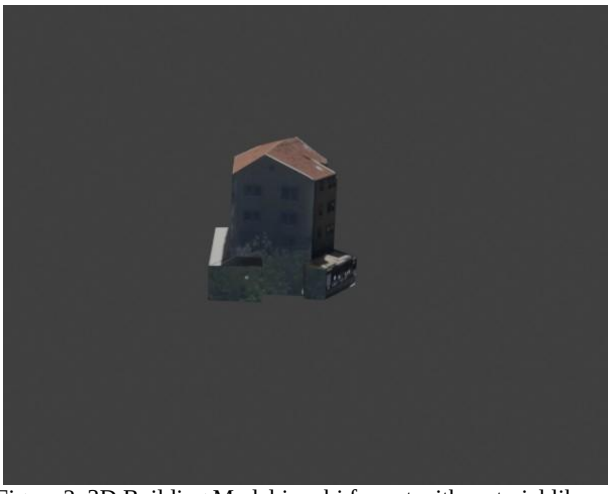

Figure 2. 3D Building Model in .obj format with material library (MTL) and textures.

Depending on project requirements, 3D models are exported in various Levels of Detail (LOD), ranging from simple block representations to highly detailed structures with intricate textures. Following export, a thorough quality assurance process ensures accuracy and performance. This involves validating alignment, identifying and correcting issues such as texture stretching, gaps, or mismapping, and optimizing OBJ files by reducing unnecessary vertices and polygons. These steps strike a balance between file size and model fidelity, ensuring the models are both lightweight and visually accurate without compromising quality.

Example of an OBJ File with MTL and Texture Integration

```
mtllib F-3017499-A.mtl
# Vertex positions
v 483721.876000 4425439.316000 875.870000
v 483721.940000 4425439.342000 875.921000
...
# Texture coordinates
vt 0.135096 0.682703
vt 0.134349 0.682137
...
# Faces with texture mapping
f 1/1 2/2 3/3
f 3/3 2/2 4/4
...
```

The MTL file referenced in this example, F-3017499-A.mtl, specifies the texture image paths and material properties. For example:

```
newmtl RoofTexture
map_Kd RoofTexture.jpg
```

(TKGM, 2025)

3. Importing and Preparing 3D Models for Unreal Engine in Blender

To ensure compatibility with Unreal Engine, OBJ files must be imported, scaled, and prepared in Blender, a versatile opensource 3D modeling software. This section elaborates on the process of importing OBJ files with their textures, scaling the models to match Unreal Engine’s requirements, and exporting the finalized models in the FBX format, detailing the rationale for this format choice over OBJ.

3.0.1 Importing OBJ Files in Blender: 242 OBJ files, along with their associated MTL and texture files (e.g., JPG or PNG), are first imported into Blender. The OBJ format provides a straightforward mechanism for transferring geometry and texture information, allowing Blender to recreate the 3D model with its defined surfaces and textures. Upon import, the model’s texture mapping is verified to ensure that the UV coordinates and material properties align correctly. If any issues arise, adjustments to the UV maps or material definitions are performed within Blender to maintain visual fidelity.

3.0.2 Scaling and Preparation for Unreal Engine: Unreal Engine imposes specific requirements for 3D models, including consistent scaling, orientation, and pivot point alignment. In Blender, the imported model is checked for its scale relative to Unreal Engine’s default unit system, where 1 Unreal Unit (UU) corresponds to 1 centimeter (Unreal Engine 5.5 Documentation, 2025c). Models are scaled and adjusted as needed to ensure they match real-world dimensions and are compatible with Unreal Engine’s physics and lighting systems. Additionally, the origin point of the model is repositioned to ensure that it aligns with the pivot point, facilitating proper placement and manipulation within the Unreal Engine environment. Normals are recalculated to avoid rendering artifacts, and material assignments are consolidated to minimize complexity. Figure 3 shows the prepared 3d building models for Unreal Engine requirements in Blender software.

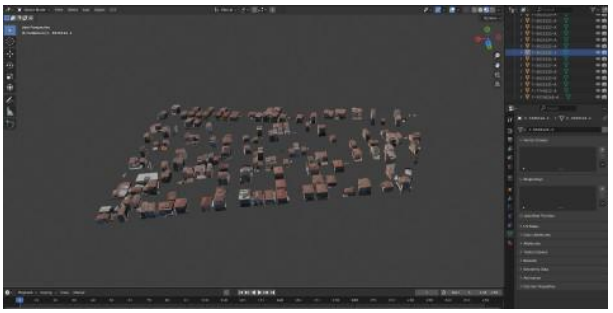

Figure 3. Scaled 3D Building Models in Blender.

3.0.3 Exporting in FBX Format: After preparation, the 3D model is exported from Blender in the FBX format. FBX is a widely used format in game development and 3D applications because of its superior capabilities compared to OBJ (Jain and Choi, 2019). Unlike OBJ, which primarily supports geometry, UV maps, and basic material definitions, FBX offers an advanced feature set, including hierarchical data, skeletal animations, and support for embedded textures. By embedding texture maps directly into the FBX file, the format streamlines the process of importing models into Unreal Engine, ensuring that textures are automatically recognized and applied during the import process. This eliminates the need for manual texture reassignment, which can be laborintensive, especially in complex scenes.

Another critical advantage of FBX is its support for animations, which makes it indispensable for dynamic 3D models requiring skeletal rigs or animated components. Although this may not be relevant for static architectural models, the capability ensures scalability for future modifications or animated elements. Furthermore, Unreal Engine natively supports FBX with optimized import pipelines, including support for advanced material properties, LODs, and collision meshes, which are either unavailable or cumbersome to implement using OBJ (Unreal Engine 5.5 Documentation, 2025a).

4. Creating a Game Project in Unreal Engine for VR Environment

The next step involves establishing a game project in Unreal Engine to develop an immersive VR environment for 3D city models. While the setup is not tied to a specific VR headset, the framework ensures compatibility with major VR platforms and headsets, providing □exibility for future deployment. This section details the creation of the project, the integration of 3D city models, and the use of Cesium for Unreal to provide a georeferenced base map environment.

4.0.1 Initial Setup in Unreal Engine: Unreal Engine (UE), a powerful and widely used game development platform, supports VR environments through its robust VR development tools, physics systems, and rendering capabilities. To begin, a new game project is created using the Blank Template, ensuring minimal overhead and maximum customization. The project settings are configured for VR compatibility, enabling features such as stereoscopic rendering, motion tracking, and input systems compatible with VR controllers.

The level editor in Unreal Engine is the primary interface for constructing the virtual environment. It provides tools for managing assets, setting up the 3D scene, configuring lighting, and adjusting physics parameters. Key components of the game environment include:

- Actors: Objects within the game environment, including3D models, cameras, lights, and geometry.
- Blueprints: Unreal Engine’s visual scripting system thatallows for rapid prototyping and functionality implementation without requiring extensive programming knowledge.
- Level Streaming: A feature used to dynamically load andunload parts of the environment, optimizing performance for large 3D city models.

(Unreal Engine 5.5 Documentation, 2025b)

4.1 Using Cesium for Unreal

To establish a georeferenced environment, Cesium for Unreal is integrated into the project. Cesium is an advanced geospatial platform that supports globally accurate coordinate systems and integrates real-world geographic data into Unreal Engine. The following steps are undertaken to configure Cesium for Unreal:

4.1.1 Installing Cesium for Unreal: The Cesium plugin is added to the project via the Unreal Marketplace. This plugin allows for seamless integration of geospatial data into the game environment.

4.1.2 Setting Up the Georeferenced World: Cesium employs the EPSG:4978 (WGS 84) geocentric coordinate reference system (CRS), which aligns with global 3D coordinates. This ensures that all imported 3D models and the base map are positioned accurately in real-world locations.

4.1.3 Adding Cesium World Terrain and Bing Maps Road:

- Cesium World Terrain: This high-resolution global terraindataset provides a realistic and georeferenced 3D

surface. It includes elevation data and natural landforms, enhancing the visual realism of the environment.

- Bing Maps Road: This basemap layer adds cartographicdetail, including roads, landmarks, and other essential geographic features, offering additional context to the 3D city models. The integration of these layers ensures that the 3D city models are accurately placed within a global geographic framework. This is particularly important for largescale urban environments where spatial accuracy and context are essential for meaningful interaction and analysis.

Figure 4 shows added Cesium World Terrain and Bing Maps Road as basemap layer in Unreal Engine.

Figure 4. Cesium World Terrain in Unreal Engine Environment.

4.2 Importing 3D City Models

The 3D city models exported in FBX format from Blender are imported into Unreal Engine as Static Meshes, a fundamental Unreal Engine asset type for non-animated objects. During import, the following parameters are adjusted:

- Location, Rotation, and Scale: The models are alignedwith the Cesium georeferenced environment using accurate coordinates.
- Collision Setup: Simplified collision meshes are added to ensure efficient physics interactions.
- Material Mapping: Textures and materials from the FBXfile are applied, ensuring the models maintain their visual fidelity.

Figure 5 shows how Unreal Engine configures FBX objects as static mesh, their materials and textures.

Although the 3D city models function correctly in local coordinate systems, their final placement into Cesium’s georeferenced scene is misaligned. The root cause is still under investigation, encompassing potential discrepancies in coordinate transforms, pivot points, or scale factors in the FBX files. Due to this unresolved issue, the subsequent steps that rely on proper geographic alignment—such as georeferenced interactions, advanced VR simulations, or integration of additional geospatial layers—are not yet fully functional.

4.3 Game Environment Configuration

In Unreal Engine, the VR Pawn or Player Character is configured to facilitate user interaction with the virtual environment. This setup involves implementing VR-compatible cameras to enable stereoscopic rendering and provide an immersive first-person view. Navigation controls, such as teleportation or smooth locomotion, are tailored for exploring the 3D city models seamlessly, ensuring user comfort and accessibility. Additionally, interaction systems are integrated to allow users to engage with the environment, such as selecting specific buildings or triggering animations to enhance interactivity.

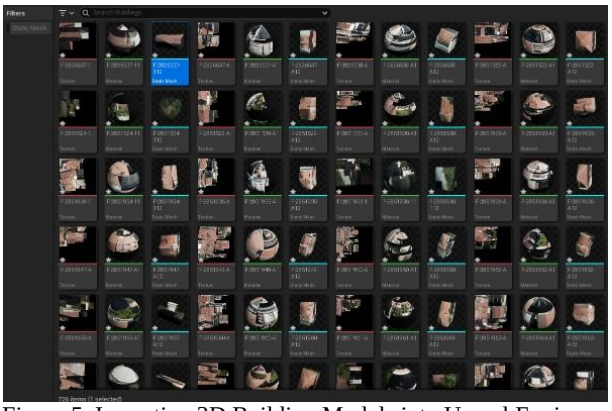

Figure 5. Importing 3D Building Models into Unreal Engine as Assets.

To create a visually compelling experience, the environment’s lighting, shadows, and post-processing effects are meticulously optimized for VR. These adjustments ensure a balance between visual quality and system performance, making the virtual space both immersive and efficient for real-time rendering.

The integration of Cesium and Unreal Engine offers significant advantages in creating immersive and geospatially accurate VR environments. Cesium’s reliance on the WGS 84 (EPSG:4978) coordinate system ensures global consistency, allowing for precise placement and scaling of 3D city models within their realworld geographic context. This geospatial accuracy is crucial for applications requiring realism and spatial reliability.

Meanwhile, Unreal Engine’s advanced rendering pipeline enhances the visual appeal of the 3D models, ensuring they are both realistic and engaging, even in the performance-intensive VR environment. Furthermore, Cesium’s extensibility enables seamless integration of additional geospatial data, such as weather layers or traffic information, enriching the environment’s functionality and making it adaptable for a wide range of interactive and analytical applications.

5. Future Work

The integration of photogrammetric 3D city models into Unreal Engine presents significant opportunities for expanding applications and enhancing user experiences in various domains. One promising avenue is the utilization of Unreal Engine’s pixel streaming technology, which enables the delivery of high-quality, real-time rendered 3D environments to web browsers and lightweight devices without requiring significant local processing power. This capability would allow seamless access to immersive 3D city models, making them available to a broader audience, including urban planners, architects, and stakeholders, regardless of their hardware limitations. Such a streaming solution would also facilitate collaborative environments where multiple users can interact with the same 3D city model remotely.

In the context of VR platforms, Unreal Engine’s robust support for different hardware, environments, and physics systems makes it an ideal tool for developing realistic and interactive simulations. Future work could involve enhancing VR interactions to allow users to not only explore 3D city environments but also manipulate and analyze them. For example, users could dynamically simulate urban scenarios such as traffic □ow, □ood modeling, or infrastructure development within a virtual cityscape. These simulations would benefit urban planning, disaster management, and public engagement by offering an

intuitive and immersive platform for understanding complex spatial data.

Further integration of GIS (Geographic Information Systems) could significantly enrich the functionality of these 3D environments. By linking geospatial data to objects within the city model, users could query buildings, roads, or landmarks for metadata such as ownership, historical significance, or construction details. This would enable a comprehensive digital twin of the urban area, supporting decision-making processes in smart city initiatives, urban sustainability planning, and infrastructure management. Incorporating real-time data streams, such as traffic conditions, weather updates, or IoT sensor feeds, could further enhance the model’s utility by providing dynamic, upto-date information for simulations and analyses.

Future advancements might also explore AI-driven features within these environments. For example, integrating machine learning algorithms could automate tasks such as feature recognition (e.g., identifying building types or land use patterns) or predictive modeling (e.g., forecasting urban growth). This would streamline processes like urban analysis, significantly reducing the manual workload required for complex studies.

Additionally, the use of augmented reality (AR) could extend the reach of these models beyond virtual environments, enabling on-site applications where city models are overlaid onto realworld landscapes. This could provide valuable tools for urban development, construction monitoring, and public engagement by merging physical and digital worlds in a seamless interface.

In conclusion, the integration of 3D city models into Unreal Engine offers a versatile and scalable platform with vast potential for future applications. By leveraging Unreal’s advanced rendering, physics, and streaming capabilities, combined with GIS integration and emerging technologies such as AI and AR, this method could redefine how 3D city models are used across disciplines, from urban planning and smart city development to public engagement and immersive storytelling.

References

Dyaksa, G. A., Arfian, N., Herianto, H., Choridah, L., Cahyanta, Y.A., 2023. Smoothing module for optimization cranium segmentation using 3d slicersmoothing module for optimization cranium segmentation using 3d slicer.

HKMO, 2005. Büyük Ölçekli Harita ve Harita Bilgileri Üretim Yönetmeliği. [Accessed: 2025-01-02].

Jain, K., Choi, Y.M., 2019. Building tangible augmented reality models for use in product development.

Jochem, R., Goetz, M., 2012. Towards interactive 3d city models on the web.

Mitishita, E.A., Habib, A., Centeno, J.A.S., Machado, M.L., Lay, J.C., Wong, C., 2008. Photogrammetric and lidar data integration using the centroid of a rectangular roof as a control point.

Parvu, I.M., Picu, I.A.C., Spiroiu, I., 2024. The importance of ground control points in a photogrammetric workflow.

Saran, S., Wate, P., Srivastav, S.K., Murthy, Y.V.N.K., 2015. Citygml at semantic level for urban energy conservation strategies.

TKGM, 2025. 3d geographical information system. Accessed: 2025-01-02.

Toledo, R., Wang, B., Levy, B., 2008. Geometry textures and applications¡sup¿†¡/sup¿.

Toutin, T., 2004. Dsm generation and evaluation from quickbird stereo imagery with 3d physical modelling.

Unreal Engine 5.5 Documentation, 2025a. Fbx content pipeline. [Accessed: 2025-01-15].

Unreal Engine 5.5 Documentation, 2025b. Level editor in unreal engine. [Accessed: 2025-01-15].

Unreal Engine 5.5 Documentation, 2025c. Units of measurement in unreal engine. [Accessed: 2025-01-15].