Combining Real-World Maps with Generative AI for Open-World Games

Tomislav Peharda, Predrag Peharda
University of Zagreb Faculty of Organization and Informatics
Artificial Intelligence Laboratory
Pavlinska 2, 42000 Varaždin, Croatia
tpeharda@foi.hr, pehardap@gmail.com

Abstract. *Large-scale virtual worlds are typically either geographically accurate but empty, or richly generated but ungrounded, transient, and difficult to edit. This paper proposes a four-layer reference architecture combining both approaches. Geographic Information System (GIS) data provides the geographic skeleton, a semantic layer converts it into a typed graph of places, and generative Artificial Intelligence (AI) enriches it with interiors, decoration, Non-Player Characters (NPCs), and location-grounded narratives. A deployment layer targets Unreal Engine, Unity, Roblox, and Unreal Editor for Fortnite (UEFN). The architecture is positioned against concurrent work on urban digital twins, autonomous-driving simulators, and gameplay world models.*

Keywords. generative AI, GIS, OpenStreetMap, open-world games, sandbox games, world generation, large language models

1 Introduction

Building large-scale virtual worlds is one of the most labour-intensive parts of video-game development. Two approaches have developed to reduce this effort. The first uses real-world data as input: OpenStreetMap (OSM) provides building footprints and road networks (OpenStreetMap Contributors, 2004), Digital Elevation Models (DEMs) provide terrain, and orbital imagery provides textures and land cover. Tools such as Arnis (E. & contributors, 2022) convert this data into Minecraft worlds at city scale. Cesium for Unreal and Unity (Cesium GS, 2025) stream globe-scale 3D content into game engines. Microsoft Flight Simulator 2024 (Microsoft & Asobo Studio, 2024) reconstructs the entire planet from Bing Maps imagery. The second approach generates the world from scratch using generative models. Genie 3 (Google DeepMind, 2025) generates navigable environments frame-by-frame from text. World Labs Marble (World Labs, 2025) generates persistent, editable 3D scenes from text, image, video, or sketch. Meta's WorldGen (D. Wang et al., 2026) targets game engines but does not ground its output in real-world geography.

The two approaches are usually framed as alternatives. This paper treats them as complementary. The GIS approach produces accurate but empty worlds. Street layouts are correct, but buildings are empty shells with no inhabitants, quests, or gameplay logic. The generative approach produces detailed worlds that are not tied to any real location. A game set in a specific real place, with generated content and characters appropriate to that place, requires both.

This paper proposes a four-layer reference architecture for the hybrid. The narrative layer (NPCs, quests, and back-stories grounded in real geography) is the main focus. This is the area least addressed by existing work and where the player-experience impact is largest.

The remainder of the paper is organised as follows. Sec. 2 reviews related work in GIS-to-game tools, generative world models, and hybrid systems. Sec. 3 presents the four-layer architecture. Sec. 4 walks through a concrete example of generating a game world for the city of Split. Sec. 5 discusses applications and open challenges. Sec. 6 concludes.

2 Related work

The GIS-to-game pipeline is mature. Arnis (E. & contributors, 2022) is written in Rust, licensed Apache 2.0, and converts OSM and elevation data into Minecraft worlds. It has over sixteen thousand GitHub stars and shipped v2.8 in May 2026. Cesium for Unreal and Unity (Cesium GS, 2025) streams 3D Tiles, photogrammetry, terrain, and imagery into game engines, and exposes a full-scale WGS84 globe to the engine runtime. Microsoft Flight Simulator 2024 (Microsoft & Asobo Studio, 2024) uses Bing Maps and cloud-side machine learning to reconstruct the planet at consumer-game fidelity. None of these systems generate content. Cesium for Unreal has no inhabitants. Arnis has no quests. Microsoft Flight Simulator has no gameplay logic. The output is a skeleton.

The generative approach combines Large Language Models (LLMs), diffusion-based image and video

generation, and world foundation models. Genie 3 (Google DeepMind, 2025) generates 720p navigable environments frame-by-frame from a text prompt. World Labs Marble (World Labs, 2025) generates persistent, editable 3D scenes from text, image, video, or rough sketches. Sora 2 (OpenAI, 2025) was a consumer-facing version of the same approach and was discontinued in April 2026. Hunyuan3D 2.0 (Zhao et al., 2025) generates game-ready textured 3D assets at scale. None of these systems takes a real-world location as input. The output is not tied to any actual place.

A body of academic work has begun to combine the two approaches, almost entirely outside the games context. The closest existing work is UrbanWorld (Shang et al., 2024), which converts an OSM-derived urban layout into a 3D scene through a four-stage LLM and diffusion pipeline, and RAISECity (S. Wang et al., 2025), a multimodal agent framework for reality-aligned 3D world generation at city scale. Both target embodied AI and urban science rather than gameplay. In the autonomous-driving simulator world, CARLA's 0.9.16 release (CARLA Team, 2025) ships a Digital Twin Tool that auto-builds environments from OSM and integrates NVIDIA Cosmos Transfer (NVIDIA, 2025) for generative scene-style variations. This is the most direct existing example of the hybrid architecture, but serves vehicle training rather than gameplay. Xu and colleagues' scoping review (Xu et al., 2024) surveys the wider Generative AI (Gen AI)-for-urban-digital-twins space. On the gameplay side, Meta's WorldGen (D. Wang et al., 2026) uses an LLM to plan scene layout but takes text prompts as input rather than geographic data. Roblox Cube (Roblox, 2025) generates functional 3D assets, also without a GIS input. Epic's State of Unreal 2025 (Epic Games, 2025) introduced the Persona Device for player-talkable AI NPCs in UEFN, but it is not connected to any GIS input. Individual components exist on both sides, but no system targets sandbox games with grounded NPCs, quests, and narrative. Table 1 summarises the systems discussed above along five dimensions: whether the system takes real-world geographic data as input, whether it targets game engines, whether it generates NPCs, whether it generates narrative content, and whether its output is a persistent artefact that can be edited or shipped with a game build.

Another set of systems targets the NPC and narrative side of games directly. Park et al. (Park et al., 2023) demonstrated that an LLM can drive a population of agents with persistent memory, daily routines, and emergent social behaviour in a small fictional town. Commercial offerings such as Inworld AI and Convai bring similar techniques to Unreal Engine and Unity as middleware for NPC dialogue. AI Dungeon and similar systems have shown that LLMs can drive interactive narrative at run time. None of these systems condition the generated NPCs or narrative on the geographic place in which they are staged. The NPCs are either generic or tied to a hand-authored fictional setting. The architecture proposed in this paper combines these dialogue and narrative techniques with the GIS grounding from the first group of related systems.

3 Proposed architecture

This paper outlines a four-layer reference architecture, depicted in Fig. 1.

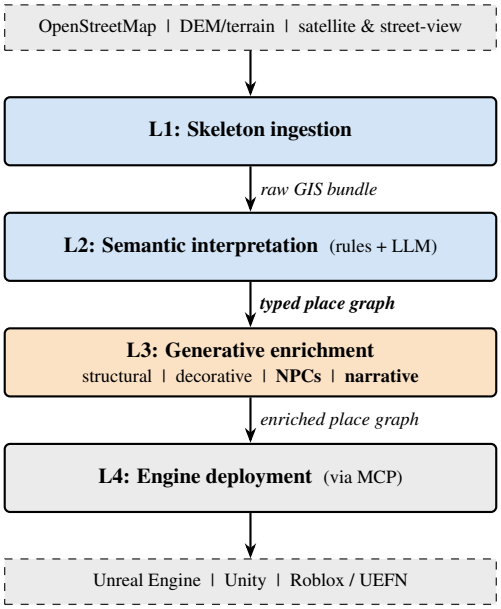


Figure 1. The four-layer reference architecture. Blue layers (L1, L2) are the deterministic GIS-grounded part; orange (L3) is the generative part; grey (L4 and I/O bands) is engine-side deployment. The L2 typed place graph is the interface between the deterministic and generative halves. Everything above it is geographic ground truth, everything below is generated.

3.1 Layer 1: Skeleton ingestion

Layer 1 ingests OSM via the Overpass API, DEMs from public sources (USGS 3DEP, Copernicus, AWS Terrain Tiles), orbital imagery (Sentinel-2, Maxar, Bing), and optionally street-view imagery for facade textures. The output is a uniformly projected and tiled raster-plus-vector bundle for the bounding region of interest. Existing libraries cover this step: Overpass-turbo for OSM queries, GDAL and rasterio for raster handling, and projects such as Arnis demonstrate that this entire stack is mature production tooling. This paper treats Layer 1 as a solved problem and as the boundary between existing tools and the proposed architecture. The architecture assumes that the input bundle contains, at minimum, a building footprint vector layer with OSM tags, a road network with classifications, a terrain raster, and a land-use raster.

Table 1. Comparison of related systems across five dimensions. ✓ indicates the system supports the feature, – indicates it does not.

System	GIS input	Targets games	Generates NPCs	Generates narrative	Persistent output
Arnis	✓	✓	–	–	✓
Cesium for Unreal	✓	✓	–	–	✓
MS Flight Sim 2024	✓	✓	–	–	✓
Genie 3	–	–	–	–	–
World Labs Marble	–	–	–	–	✓
Meta WorldGen	–	✓	–	–	✓
Roblox Cube	–	✓	–	–	✓
UEFN Persona Device	–	✓	✓	–	✓
UrbanWorld	✓	–	–	–	✓
RAISECity	✓	–	–	–	✓
CARLA + Cosmos	✓	–	–	–	✓
This paper	✓	✓	✓	✓	✓

3.2 Layer 2: Semantic interpretation

Raw GIS data is geographically accurate but has limited semantic content. An OSM `building=yes` tag indicates that a polygon is a building, but not the type of building, the era it was built in, or the activities that take place inside it. Layer 2 converts the raw data into a *typed place graph*. Nodes are typed entities such as a primary school, a residential road, or an Italian restaurant. Edges describe spatial and functional relations such as adjacency, access, or proximity. The interpretation uses rules (OSM tag schemas, land-use polygons) and an LLM that fills in gaps and resolves ambiguity.

The rule layer handles the cases where OSM tags are unambiguous: a polygon tagged `amenity=school` becomes a school node, a way tagged `highway=residential` becomes a residential road edge. The LLM layer handles the more common case of missing or ambiguous tags. For example, an unlabelled rectangular polygon next to a labelled school in a residential area is more likely a sports field than a parking lot. The LLM is prompted with the candidate polygon and its neighbourhood context, and produces a typed label with a confidence score. Labels below a confidence threshold are flagged for designer review.

The typed place graph is the interface between L2 and L3. Everything above this line is geographic ground truth, everything below is generated.

3.3 Layer 3: Generative enrichment

Layer 3 contains the generative parts of the architecture. Its outputs fall into four categories: structural, decorative, populated, and narrative.

Structural enrichment. Building interiors are generated by an LLM conditioned on the building type, era, and the player-supplied style prompt, with the constraint that the generated floor plan fit within the footprint provided by L2. The number of floors is taken from OSM `building:levels` where available and inferred from height tags or street-view imagery otherwise. Interior assets are drawn from a retrieval-augmented library (Lewis et al., 2020) of pre-authored components and supplemented by text-to-3D generation (Zhao et al., 2025) when no library entry fits. The output of this step is a per-building floor plan and asset list, attached to the corresponding place graph node.

Decorative enrichment. Vegetation is sampled from species distributions appropriate to the geographic latitude and biome of the target region, which can be derived from the terrain raster and global biome datasets. Street furniture (benches, lampposts, hydrants) is placed deterministically where OSM provides the data and probabilistically where it does not, using density priors per land-use class. Facade textures, when not derivable from street-view imagery, are generated by a diffusion model conditioned on a regional style prompt and the era tag of the building.

Populated enrichment. Each typed place node receives a population pass that instantiates one or more NPCs appropriate to its type. A school receives teachers and students. A restaurant receives staff and patrons. A residential building receives household members. The number and type of NPCs per node follows configurable density priors that can be adjusted per region or per game type. NPCs are generated with persistent identities, life-histories, and behavioural patterns, following the generative-agents approach (Park et al., 2023). The difference is that the identities are conditioned on the real place: name, history, and dialect appropriate to the region. The LLM prompt for an NPC is given both the local node context (a bakery owner in a coastal Mediterranean city) and the regional context (Adriatic coast, local dialect area), allowing consistent voice and background across all NPCs in a region.

Narrative enrichment. Quests, fetch tasks, conversations, and lore are generated against the place graph. Two types of grounding are supported. Historical grounding draws on the documented history of the area, for example a quest set in Diocletian's Palace in Split with real Roman-era context. Synthetic ground-

ing generates fiction consistent with the affordances of the place, for example a heist in a real bank or a quest taking place in a real plaza. The geographic grounding is, to our knowledge, the main novelty relative to prior LLM-driven NPC and narrative systems.

3.4 Layer 4: Engine deployment

Layer 4 targets one of four commercial engines and platforms: Unreal Engine, Unity, Roblox, and UEFN. For each target, the enriched graph is compiled into the native asset bundle and NPC runtime. The deployment is engine-specific but architecturally uniform. The interface at the L3-L4 boundary is the enriched graph itself, which is engine-agnostic. Compilation on the Unreal side uses Cesium for Unreal for the streamed globe and converts the enriched-graph assets into native Actors. On the Roblox side, the same graph compiles to Roblox Studio scripts and Cube-generated 3D assets. The UEFN Persona Device (Epic Games, 2025) and Roblox Cube (Roblox, 2025) provide engine-side runtimes for generative NPCs and assets respectively. Custom socket protocols used in earlier systems are replaced by the Model Context Protocol (MCP) (Anthropic, 2024) as the interface between models and engines. Runtime model calls, for example during a player conversation with an NPC, are routed through the same MCP interface, which allows the model provider to be swapped without engine-side changes.

4 Example scenario

This section walks through an end-to-end example: generating a small open-world game set in the historical centre of Split, Croatia. The bounding region is roughly 1 km by 1 km around Diocletian's Palace, the Roman-era core of the city. The target engine is Unreal Engine.

Layer 1. An Overpass API query retrieves all OSM features within the bounding box: roughly 500 building footprints, the road network, parks, the waterfront promenade, and tagged points of interest. DEMs are pulled from Copernicus at 10 m resolution. Sentinel-2 imagery provides the satellite raster and ESA WorldCover provides land-use classification. Optionally, Mapillary street-view imagery is queried along the road network for facade reference.

Layer 2. The interpretation step produces a typed place graph with about 500 building nodes, 100 road segment edges, and around 80 Point of Interest (POI) nodes. OSM tags carry the obvious cases directly: Diocletian's Palace is tagged `historic=palace`, the Cathedral of Saint Domnius is tagged `building=cathedral`, the Peristyle is tagged as a public square. The LLM resolves the remaining cases. Untagged polygons within the palace walls are labelled as residential or commercial buildings of the period based on their footprint and neighbourhood. Untagged polygons along the waterfront are labelled as cafes, restaurants, or shops based on the surrounding land-use class. Fig. 2 shows a small subgraph produced by this step.

Layer 3. For structural enrichment, Diocletian's Palace gets a generated layout consistent with late-Roman imperial architecture, the cathedral gets a nave-and-side-chapel layout adapted from the original mausoleum, and residential buildings inside the old town get tight apartment floor plans appropriate to Mediterranean urban housing. For decorative enrichment, the parks and palace courtyards are populated with cypress, olive, and pine trees characteristic of the Mediterranean biome, street furniture is placed along the waterfront and pedestrian zones, and facade textures are generated in the Roman-to-Venetian-revival range typical of the city centre. For populated enrichment, the cafes along the Riva receive baristas and patrons, the palace receives guides and visitors, the cathedral receives clergy and tourists, and residential buildings receive household-scale populations. NPC identities use Croatian names and references to the Chakavian dialect of the Adriatic coast. For narrative enrichment, the system generates quests that reference the documented history of the city. Diocletian's Palace was built around 305 AD as a retirement residence for the Roman emperor Diocletian. The city of Split grew within and around its walls after the Avars and Slavs destroyed nearby Salona in the seventh century. Quests can be staged around any of these periods. Smaller quests are generated for individual locations, for example a fetch task involving a fish vendor on the Riva or a conversation with a guide at the Peristyle.

Layer 4. The enriched graph is compiled into an Unreal Engine project. The globe is streamed via Cesium for Unreal, the generated assets are imported as Actors with their place-graph identity preserved as Actor metadata, and the NPC runtime is wired through the MCP interface so that conversations are served by an LLM at play time. The resulting project is a playable Unreal scene in which the player can walk through a recognisable Split and interact with NPCs whose identities and dialogue are grounded in the actual city.

Sample player experience. A player spawns on the Peristyle. The surrounding cafes are populated with NPCs drinking coffee. The player approaches one of the fish vendors on the Riva, who uses a Croatian name and refers to her family's history in the fishing trade. She offers a fetch quest involving a delivery to the cathedral. As the player walks toward the cathedral through the narrow palace streets, the geography matches the real layout, the buildings have appropriate facades, and other NPCs along the way react to the player's progress. The quest line, the NPCs the player meets, and the locations they visit are all generated. The spatial structure and the historical and cultural context match the real city. Re-running the generation with the same seed produces the same world, allowing the

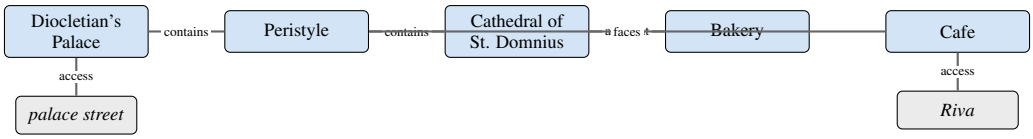


Figure 2. A subgraph of the L2 typed place graph for the Split example. Blue nodes are typed place entities; grey nodes are roads. Edges are typed relations (contains, adjacent, faces, access).

game to be shipped as a deterministic artefact. Fig. 3 summarises the flow of the example scenario across the four layers.

5 Discussion

GIS-only worlds are accurate but static and uninhabited. Purely generative worlds are detailed but not tied to any real location. A hybrid uses GIS for spatial accuracy and generative AI for content. This allows a game to be set in a real place without manual authoring of every detail. A game set in the player's hometown becomes feasible without a dedicated content team per town. A historical reconstruction (ancient Rome, medieval Dubrovnik, eighteenth-century Edinburgh) can be assembled from the same architecture by changing the era prompt and the historical sources fed into the narrative layer. Location-based games can stage encounters that fit the specific place, rather than placing generic content at arbitrary coordinates.

The hybrid keeps the editability and persistence that pixel-space generative world models lack. The enriched place graph is a typed, serialised artefact that ships with the game build. It is editable by designers and inspectable by audit tools. The compute cost is paid once per region, not per frame of play.

Several open challenges remain.

Cultural sensitivity. Generating content set in real places risks producing insulting or inaccurate content about real communities and sacred sites. Religious buildings, sites of historical trauma, and contested borders all require treatment that current LLMs are unreliable at delivering without strong guardrails. A region-specific editorial layer is a prerequisite for any deployment beyond a research prototype. For sensitive cases, involvement of local communities is also required.

Multiplayer consistency. If two players visit the same real place at different times or on different clients, the generated content should be substantially the same. This is straightforward when the generators are deterministic functions of (place, seed, version), as in the Split example above. It is harder when remote model providers introduce non-reproducible sampling, and harder still when generated content evolves over time as players interact with it. The architecture must include a content versioning mechanism that allows two clients to agree on the version of the world they are experiencing.

Scale. Generating per-building interiors at city scale is computationally prohibitive without a tiered strategy. A practical deployment would generate full-fidelity interiors only for buildings the player is near, lower-fidelity placeholders for buildings in the visible distance, and only the typed place graph node for everything further away. Smooth transitions between tiers and pre-generation of high-traffic areas are open problems.

Asset reuse economics. Generating each interior from scratch is wasteful when most buildings of a given type are similar. A more economical strategy is to compose interiors from a curated library of typed components and perturb each instance slightly. This requires a substantial library to exist before the architecture works well at scale.

Legal status. A game set in a real city raises questions about likeness rights, trademarks (a real coffee chain's logo on a real corner), and the depiction of real inhabitants by NPCs that resemble them. These are legal rather than technical questions, but they shape what the architecture can be permitted to do in a commercial product.

6 Conclusion

Sandbox and open-world games can benefit from combining the GIS skeleton approach and the generative-detail approach. The intersection has been explored for urban science and autonomous-vehicle training, but not for gameplay. This paper has outlined a four-layer reference architecture that combines GIS ingestion, semantic interpretation, generative enrichment, and engine deployment. Grounded NPCs and narratives are highlighted as the main novelty. Future work will turn the proposal into an implementation. The narrative layer is the priority, as it is the least explored. We will also conduct user studies of designer-facing and player-facing variants of the architecture.

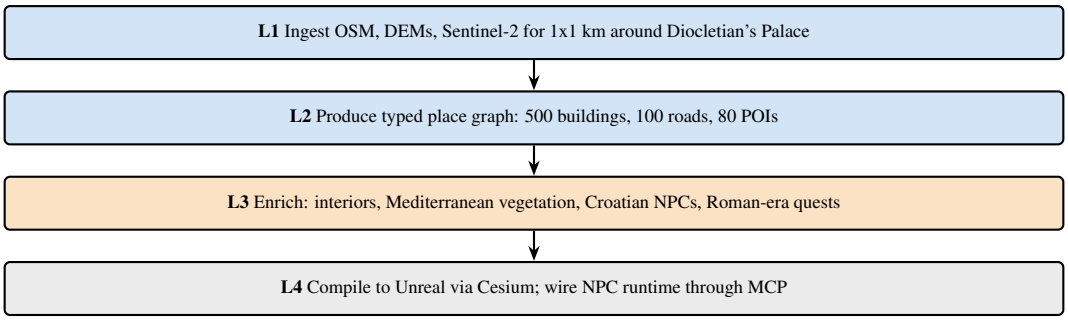


Figure 3. The four-layer pipeline applied to the Split example scenario, from raw GIS ingestion to a playable Unreal Engine build. Colours match Fig. 1.

References

Anthropic. (2024, November). Introducing the model context protocol.

CARLA Team. (2025, September). CARLA 0.9.16: Digital twin tool, NVIDIA Cosmos Transfer and NuRec integration.

Cesium GS. (2025). Cesium for Unreal and Unity: 3d geospatial streaming for game engines.

E., L., & contributors. (2022). Arnis: Generate minecraft worlds from real-world locations [Apache 2.0; uses OpenStreetMap and AWS Terrain Tiles. Accessed: 2026-06-07].

Epic Games. (2025). State of Unreal 2025: Persona device, scene graph, developer assistant.

Google DeepMind. (2025, August). Genie 3: A new frontier for world models.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, *33*, 9459–9474.

Microsoft & Asobo Studio. (2024). Microsoft flight simulator 2024.

NVIDIA. (2025). Cosmos: World foundation models for physical AI.

OpenAI. (2025, September). Sora 2 [Discontinued April 2026].

OpenStreetMap Contributors. (2004). OpenStreetMap.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. *Proceedings of the 36th annual acm symposium on user interface software and technology*, 1–22.

Roblox. (2025, March). Introducing Roblox cube: A foundation model for 3d intelligence.

Shang, Y., Lin, Y., Zheng, Y., Fan, H., Ding, J., Feng, J., Chen, J., Tian, L., & Li, Y. (2024). Urbanworld: An urban world model for 3d city generation. *arXiv preprint arXiv:2407.11965*.

Wang, D., Jung, H., Monnier, T., Sohn, K., Zou, C., Xiang, X., Yeh, Y.-Y., Liu, D., Huang, Z., Nguyen-Phuoc, T., et al. (2026). Worldgen: From text to traversable and interactive 3d worlds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 27124–27135.

Wang, S., Zheng, Z., Shang, Y., He, L., Yu, Y., Hangyu, F., Feng, J., Liao, Q., & Li, Y. (2025). Raisecity: A multimodal agent framework for reality-aligned 3d world generation at city-scale. *arXiv preprint arXiv:2511.18005*.

World Labs. (2025, November). Marble: Generate, edit, and inhabit persistent 3d worlds.

Xu, H., Omitaomu, F., Sabri, S., Zlatanova, S., Li, X., & Song, Y. (2024). Leveraging generative ai for urban digital twins: A scoping review on the autonomous generation of urban data, scenarios, designs, and 3d city models for smart city advancement. *Urban Informatics*, *3*(1), 29.

Zhao, Z., Lai, Z., Lin, Q., Zhao, Y., Liu, H., Yang, S., Feng, Y., Yang, M., Zhang, S., Yang, X., et al. (2025). Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*.