The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume L-4/W1-2026
Free and Open Source Software for Geospatial (FOSS4G) 2026 – Academic Track "Bridging Geospatial Technology and Humanity",
30 August–5 September 2026, Hiroshima, Japan

From CityGML to Game Engine: A Reproducible Open-Source GIS Pipeline for Interactive 3D Urban Environments Using PLATEAU, Blender and Godot

Léo Martial [1], Hiroyuki Kubono [1], Shino Miura [1]

[1] Faculty of Science and Engineering, Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan
lmartial960@g.chuo-u.ac.jp

Keywords: CityGML, PLATEAU, Godot Engine, Open-Source GIS, 3D City Models, Reproducible Pipeline

Abstract

Open 3D city-model datasets are increasingly available worldwide through the CityGML standard, and Japan's Project PLATEAU is among the largest national open initiatives of its kind. Yet official software development kits exist only for the proprietary Unity and Unreal engines, and no documented route connects PLATEAU data to an open-source interactive environment. This paper presents and evaluates a fully open-source geospatial pipeline that transforms CityGML data into a lightweight, interactive, real-time 3D application suitable for the low-specification hardware typically found in schools and community centres. The pipeline relies exclusively on free and open-source software, using Blender for geospatial processing and 3D optimisation and the Godot Engine for real-time rendering and interaction. We document the pipeline at the level of detail required for reproduction, including the PLATEAU data model and tiling scheme, the reprojection from the native geographic frame (EPSG:6697) to a metric local frame, the geometry-repair and level-of-detail decisions, and the automatic generation of the collision geometry that a game engine requires but a GIS does not. The pipeline was validated as a deployment stress test in a participatory urban-design workshop with approximately fifty elementary-school students in Yokohama, during which 1,192 assets were placed without system failure on six standard classroom computers. We identify Blender as the indispensable bridge component and the collision layer as the technical linchpin, and we argue that the present manual workflow, while reproducible, is still too complex for non-experts; we therefore outline a realistic turnkey, automated successor.

1. Introduction

Open three-dimensional city-model datasets are becoming a foundational layer of the contemporary geospatial commons. Encoded in the Open Geospatial Consortium standard CityGML, these datasets describe buildings, terrain, vegetation and infrastructure as semantically structured objects rather than as undifferentiated meshes. In Japan, the Ministry of Land, Infrastructure, Transport and Tourism has released such models at national scale through Project PLATEAU, an open initiative that now covers more than two hundred municipalities and publishes its data under open licences (Ministry of Land, Infrastructure, Transport and Tourism, 2024; Seto et al., 2023).

These datasets are increasingly used beyond cartography. Game engines have become attractive platforms for participatory planning, education and civic engagement because they combine real-time rendering, intuitive navigation and physics-based interaction. PLATEAU, however, provides official software development kits only for the proprietary Unity and Unreal engines. For practitioners committed to a free and open-source toolchain, or who must deploy on the modest hardware available in classrooms and community centres, no documented path leads from CityGML to a fully interactive environment. The geospatial community therefore lacks a reproducible pipeline for turning CityGML data into lightweight, interactive 3D applications using open-source software alone.

This paper addresses that gap from a geospatial-engineering perspective. We present and document a complete pipeline from CityGML acquisition to an interactive real-time 3D environment, built exclusively with free and open-source software: Blender for geospatial data processing and 3D optimisation, and the Godot Engine for real-time rendering and interaction. The contribution is threefold. First, we provide the first documented end-to-end workflow for integrating CityGML and PLATEAU data into the Godot Engine by way of Blender, filling the gap left by the absence of an official open-source software development kit. Second, we document, at a level of detail sufficient for reproduction, the geospatial obstacles that arise when geometry intended for visualisation is repurposed for physics-based interaction, namely coordinate handling, geometry repair, level-of-detail management and collision generation. Third, we examine the usability cost of the workflow and outline a realistic automated successor that would make it accessible to non-experts.

The present paper is deliberately focused on the pipeline. The workshop is treated here only as a deployment stress test that exercises the pipeline under real classroom constraints; the spatial analysis of the placement data is reported in a companion study (Martial et al., 2026a), and the pedagogical evaluation of the workshop in a second companion study (Martial et al., 2026b), with the underlying participatory framework developed in earlier work (Martial et al., 2025). Section 2 reviews open 3D city models and the use of game engines as planning tools. Section 3 describes the four-stage pipeline in detail. Section 4 documents the cross-cutting technical challenges. Section 5 defines a performance-characterisation methodology for low-specification hardware. Section 6 reports the validation. Sections 7 and 8 discuss the findings and set out an automated successor to the pipeline.

2. Background and related work

2.1 Open 3D city models and PLATEAU

CityGML is the international standard for the representation and exchange of semantic 3D city models. It organises the urban

scene into thematic modules, including buildings, relief, vegetation and transportation, and supports several Levels of Detail (LOD), from the extruded blocks of LOD1 to the differentiated roof shapes and facade surfaces of LOD2. The most recent revision, CityGML 3.0, separates the conceptual model from its encodings and aligns it with the ISO 191xx family of standards, broadening the range of supported applications (Kolbe et al., 2021; Kutzner et al., 2020). Beyond geometry, CityGML attaches semantic attributes to each object, so that a building is an entity with a function, a height and a construction context rather than a bare surface. An interactive pipeline consumes only a fraction of this richness, but its presence means the same source could later drive attribute-based behaviour, such as colouring buildings by use.

Project PLATEAU adopts CityGML as its application schema and publishes its 3D city models as open data, with open-source converters to common GIS formats (Seto et al., 2023; Project PLATEAU, 2024). The breadth of the initiative, together with its open licensing, makes it an ideal substrate for experimentation; it also makes the absence of an open-source interactive pathway all the more conspicuous.

2.2 Game engines as planning and authoring tools

The potential of game engines for landscape and environmental visualisation was recognised more than two decades ago, and the democratisation of engines such as Unity and Unreal extended their reach into architectural and planning practice. The integration of georeferenced data was a further step, allowing engines to render real-world environments. That game environments are not neutral depictions of space, but cultural artefacts that shape how players read territory, has been argued at length elsewhere (Adams, 1998; Ash and Gallacher, 2011; Martial, 2019), and we do not revisit it here; the present account is concerned with the geospatial engineering rather than the representation politics of the medium.

The Godot Engine, first released in 2014 under the permissive MIT licence, is widely regarded as the first comprehensive and accessible open-source game engine; its lightweight architecture and its Python-like scripting language, GDScript, make it well suited to educational and participatory applications (Holfeld, 2024; Linietsky et al., 2025). Despite this, Godot is absent from the list of engines for which PLATEAU offers a software development kit.

2.3 The open-source integration gap

The consequence of this absence is concrete. A practitioner who wishes to bring PLATEAU data into Unity or Unreal can follow an officially supported import path; a practitioner who wishes to use an open-source engine has no comparable guidance and must assemble a workflow from general-purpose tools. The difficulty is not the rendering, which game engines handle well, but the chain of preparatory geospatial steps that turn analysis-oriented CityGML geometry into an interactive scene: format conversion, coordinate handling, geometry repair, level-of-detail management and, above all, the generation of the collision geometry that a game engine requires but a GIS does not. This paper treats that chain as the object of study. Two findings organise the account: the role of Blender as the bridge component, and the role of collision geometry as the technical linchpin.

3. Pipeline architecture

The pipeline comprises four sequential stages, summarised in Figure 1. Each stage relies solely on free and open-source software, and each produces an intermediate artefact that the next stage consumes. The following subsections describe the stages in turn, using the workshop site as a running example.

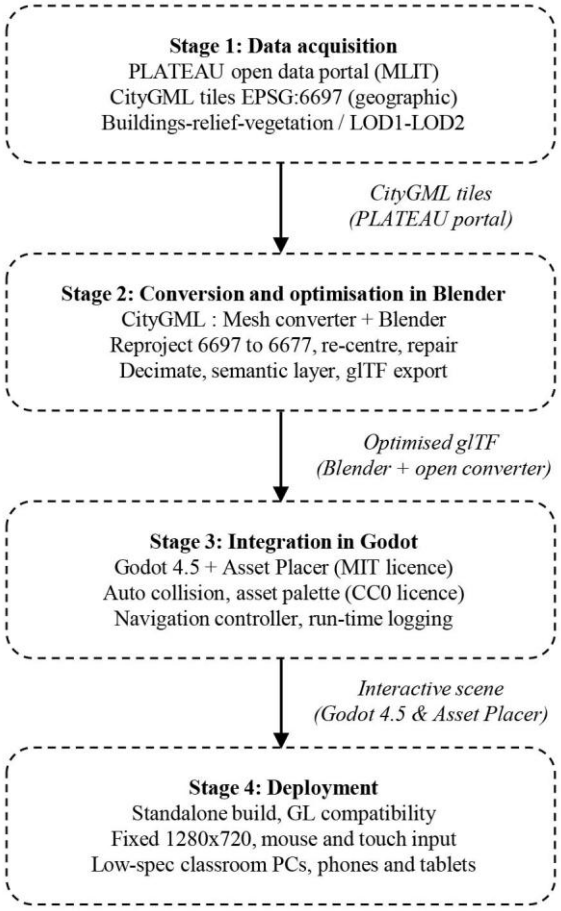

Figure 1. The four-stage open-source pipeline, from PLATEAU CityGML acquisition to an interactive Godot environment.

3.1 Stage 1: Data acquisition

The target site was Takashima Central Park, in the Minato Mirai district of Yokohama, an approximately rectangular area of about 113 m by 159 m. Acquisition begins not with a single file but with an understanding of how PLATEAU organises its data. Each municipal dataset is distributed as a collection of CityGML files partitioned by the Japanese standard regional grid, so that the city is split into mesh tiles identified by a numeric mesh code. Within a tile, the data are separated by thematic module into subfolders, typically buildings, relief (terrain), vegetation and transportation, and each file name encodes the mesh code, the theme, the coordinate reference system and the source level of detail, following the pattern mesh_theme_6697_LOD_op.gml. Textures, where present for LOD2, are stored in a sibling appearance folder.

Two properties of this distribution shape the rest of the pipeline. First, the native coordinate reference system is EPSG:6697 (JGD2011 with ellipsoidal height), a geographic system expressed in degrees rather than metres; PLATEAU adopts it precisely because no single plane-rectangular zone covers the whole country. The metric coordinates that a game engine needs

must therefore be produced by reprojection downstream, not read directly from the file. Second, the data can be very large: dense central wards run to several gigabytes compressed, so tile selection is a meaningful first decision rather than a formality.

Acquisition therefore reduced to three operations: selecting the mesh tiles that cover the site from the PLATEAU open-data portal, downloading the relevant thematic modules in CityGML, and recording the source level of detail of each layer for use in conversion. The choice of LOD involves a critical trade-off. LOD2 provides roof shapes and facade detail and so improves recognisability, but it substantially increases the polygon count and therefore the rendering cost. A selective strategy was adopted: LOD2 was retained for the buildings within the workshop perimeter, where visual fidelity matters most, and simplified representations were used for the surrounding context. Only the geometric content and the broad thematic distinction between terrain, buildings and vegetation were propagated downstream; the richer attribute set was not required for this application, although preserving it would be straightforward and would enable attribute-driven styling in later versions.

3.2 Stage 2: Conversion and optimisation in Blender

Stage 2 is where most of the geospatial work happens, and where the open-source pathway diverges most sharply from the proprietary one. CityGML cannot be imported into Blender directly, so a conversion to a mesh interchange format is required first. The routes documented by PLATEAU are uneven from an open-source standpoint: the official software development kits export OBJ, FBX and glTF but run inside Unity or Unreal; the commercial FME platform is capable but paid. The fully open alternatives are the PLATEAU GIS Converter, an open-source tool that reads CityGML and writes OBJ and other geospatial formats (Project PLATEAU, 2024), and general-purpose libraries such as GDAL/OGR for the planar data (GDAL/OGR contributors, 2025). We used an open CityGML-to-mesh conversion of this kind to obtain a textured OBJ, which Blender then imports. We deliberately treat glTF (Khronos Group, 2021), an open Khronos standard, as the interchange format out of Blender, rather than the proprietary FBX, so that no closed format appears anywhere in the chain even though Blender can read both.

Inside Blender the geometry is brought into a usable state through five operations. The first is coordinate handling, detailed in Section 4.1 and illustrated in Figure 2: the model is reprojected from the geographic EPSG:6697 frame to the metric EPSG:6677 frame (JGD2011 / Japan Plane Rectangular CS IX, which covers Yokohama) and then translated close to the scene origin. The second is geometry repair, removing degenerate faces, correcting inverted normals and resolving non-manifold configurations introduced by the conversion. The third is mesh simplification: decimation is applied with a per-layer budget so that visually important surfaces inside the perimeter retain more detail than background context, which keeps the polygon count within the rendering envelope of classroom hardware. The fourth is semantic layering: the cleaned geometry is grouped into named collections corresponding to terrain, buildings, vegetation and public space, mirroring the thematic structure of the source CityGML, which both allows the per-layer simplification budgets and gives the engine a meaningful scene hierarchy instead of a single monolithic mesh. The fifth is material consolidation: the many materials emitted by the converter are merged into a small palette to limit draw calls, a decisive factor for the integrated-graphics hardware targeted at deployment. The processed model is exported in glTF, which preserves the metric coordinate alignment and so keeps the geometry registered to its real-world position.

3.3 Stage 3: Integration in Godot

The optimised glTF model was imported into the Godot Engine, version 4.5. The first and essential step was the automatic generation of collision shapes for the imported meshes, a requirement absent from conventional GIS workflows but central to a game engine, discussed in Section 4.5 and illustrated in Figure 3. The render backend was set to GL Compatibility so that the project would run on integrated graphics, the viewport was fixed at 1280 by 720 pixels, and input was handled through two parallel schemes, mouse-and-keyboard and touch with on-screen virtual joysticks, so that one build could be operated on desktops, laptops and tablets alike.

Asset placement was provided by the Godot Asset Placer, a free editor plugin distributed under the MIT licence (Levinzon, 2024), which presents assets as a palette and places the selected object where the user clicks, snapping it onto the nearest collision surface by ray casting. The contribution of the present work at this stage is the curation rather than the plugin: a small library of freely licensed CC0 objects from the Kenney collection (Kenney, 2024) was prepared, each as a self-contained model carrying its own baked collision shape, and organised into a few colour-coded functional categories. Constraining the palette in this way gave non-expert participants an intuitive, recombinable vocabulary appropriate to a short workshop. The exact composition of the library and the analysis of how participants used it are reported in the companion study (Martial et al., 2026a).

Navigation was provided by a third-person controller with an orbital camera following a visible character at pedestrian scale, a deliberate choice for an audience of children: a third-person view gives a clearer reading of space and proportions and reinforces the playful character of the activity, whereas a first-person controller is better suited to adult workshops that aim to convey the lived pedestrian experience. Finally, an embedded GDScript routine writes a per-group placement log at run time, which is relevant here only as evidence that the engine's scripting layer can double as a lightweight geospatial data-collection instrument; the log schema and the spatial analysis it supports are detailed in the companion study (Martial et al., 2026a).

3.4 Stage 4: Deployment

The environment was packaged as a standalone Godot project deployable on standard educational computers without dedicated graphics hardware. The GL Compatibility renderer, the conservative viewport and the dual input scheme together kept the runtime requirements within reach of machines without a discrete graphics card. Godot's single-codebase export model means that the same project can be packaged for the major desktop operating systems and, in principle, for the web, without changes to the application logic; for the workshop a desktop build was used on the school computers. The deployment target imposed the practical constraints of the classroom: stable frame rates and responsive interaction had to be maintained throughout sessions run by non-expert users.

4. Technical challenges and solutions

Repurposing CityGML geometry, which is designed for visualisation and spatial analysis, into an interactive game environment raises several challenges that deserve

documentation, because documenting them lowers the barrier to entry for future users of the pipeline.

4.1 Coordinate reference system handling

As noted, PLATEAU CityGML is delivered in the geographic frame EPSG:6697, in degrees. For an interactive scene the data must be in metres, so the first transformation is a reprojection to a projected metric frame; for Yokohama this is EPSG:6677, JGD2011 / Japan Plane Rectangular CS IX. After reprojection the model carries very large coordinates, of the order of tens of kilometres from the zone origin, which degrades single-precision floating-point arithmetic and complicates interaction. A re-centring translation in Blender then moves the model close to the scene origin while preserving the internal metric distances. Because the re-centring is a pure translation, it is exactly reversible, so coordinates extracted from the engine can be mapped back to real-world positions. The full sequence, geographic to projected to local, is shown in Figure 2.

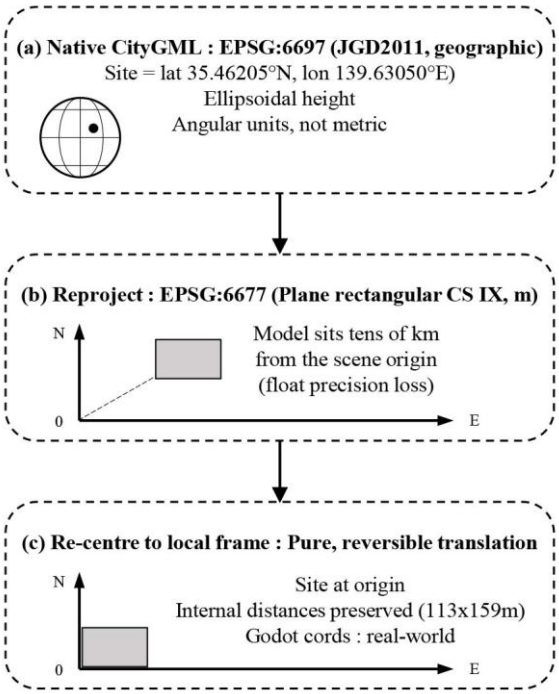

Figure 2. Coordinate handling: the model is reprojected from the native geographic frame (EPSG:6697) to a metric frame (EPSG:6677), then re-centred near the origin. Internal distances are preserved and the mapping is reversible.

4.2 Conversion artefacts

The conversion from CityGML to a mesh format introduced recurrent artefacts: flipped normals, Z-fighting on coplanar surfaces, and disconnected mesh fragments. These required manual inspection and correction in Blender. None is individually difficult to resolve, but their cumulative effect can discourage a newcomer; documenting the typical artefacts and their remedies is itself a contribution to reproducibility.

4.3 Level of detail and performance

Retaining full LOD2 for every building exceeded the rendering capacity of classroom-grade computers. The selective LOD strategy of Section 3.1, combined with the per-layer decimation of Section 3.2, kept frame rates interactive while preserving enough spatial realism for participants to recognise the site. The quantitative characterisation of this trade-off is the subject of Section 5.

4.4 Absence of an official Godot SDK

Unlike Unity and Unreal, Godot has no PLATEAU software development kit. Blender proved to be the component that makes the open-source pathway viable, acting simultaneously as format converter host, geometry optimiser and semantic structurer. It is, in effect, the bridge component of any open-source CityGML-to-game-engine pipeline, and its role should not be treated as incidental.

4.5 Collision generation

CityGML meshes are authored for visualisation and analysis, not for physics-based interaction. A game engine, however, requires collision shapes: without them a character controller passes through walls and terrain, and, equally important, the ray-cast placement workflow has no surface onto which to snap objects. Automatic generation of collision shapes from the imported meshes was effective for terrain and building shells but required verification for thin or geometrically complex elements, where collision approximations can produce either unintended barriers or gaps. The collision layer thus serves a dual purpose, supporting both navigation and placement, which is why it is treated here as the technical linchpin of the integration stage (Figure 3).

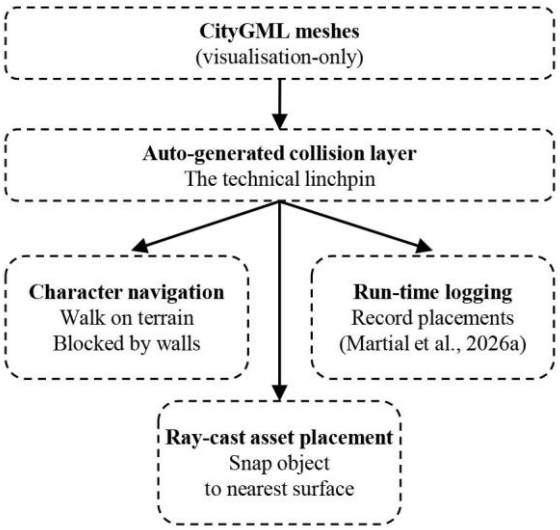

Figure 3. The automatically generated collision layer is the technical linchpin of integration: it is derived on import from analysis-oriented CityGML meshes and underpins navigation, ray-cast asset placement and run-time logging alike.

4.6 Reproducibility of the toolchain

Every component of the pipeline is openly licensed, which is what makes the workflow reproducible rather than merely described. Godot is distributed under the MIT licence and Blender under the GNU General Public Licence; the Asset Placer plugin is MIT-licensed; the Kenney library is released under CC0; the conversion tooling is open source; and the PLATEAU source data are published as open data. A reader can therefore assemble the identical toolchain at no cost, obtain the same input

data, and reproduce the procedure end to end, in contrast to pipelines that depend on a proprietary engine or paid assets.

5. Performance characterisation methodology

Because the pipeline targets low-specification hardware, the rendering cost must be characterised rather than asserted. This section defines the measurement methodology; the quantitative values are being collected on representative classroom hardware and will be reported with the corresponding measurements. The criterion adopted is not a peak frame rate on a powerful machine but sustained, responsive interaction on the weakest device in a typical classroom fleet, which is the practical question a practitioner faces before a workshop.

5.1 Metrics

Three families of indicators are recorded: geometric complexity (triangle and vertex counts per semantic layer, before and after simplification, for each LOD configuration); rendering performance (sustained mean frame rate and the first-percentile low frame rate under scripted interaction); and resource use (graphics and system memory footprint, project load time and executable size). Each metric is reported against the number of placed assets, so that the scaling behaviour of the scene under heavier user-generated load can be assessed.

5.2 Hardware and protocol

Measurements are taken across the range of devices found in schools, namely a desktop classroom computer with integrated graphics, a low-cost laptop and a tablet, rather than on a single high-end machine, all using the GL Compatibility backend. A fixed, scripted navigation path traverses the site so that frame-rate sampling is reproducible and independent of operator behaviour, and is repeated at increasing placed-asset counts, from an empty scene up to and beyond the heaviest design observed during the workshop. At a qualitative level, the workshop itself already provides a lower bound: the environment ran without failure and remained interactive on six standard classroom computers under sustained use, which the formal measurement is intended to quantify.

6. Validation: deployment in a participatory workshop

The pipeline was validated as a deployment stress test in a participatory urban-design workshop conducted on 17 December 2025 at an elementary school in Yokohama. Approximately fifty sixth-grade students worked in twelve groups across two sessions on six standard classroom computers, redesigning the park within the interactive environment produced by the pipeline. Over the two sessions the participants placed 1,192 assets without any system failure, which demonstrates the robustness of the pipeline under real deployment constraints and in the hands of non-expert users. The embedded logging routine generated a complete per-group placement record, confirming that the engine's scripting layer can function as a lightweight data-collection instrument rather than only as a rendering surface. The detailed spatial analysis of these records is reported in Martial et al. (2026a) and the pedagogical evaluation in Martial et al. (2026b); the present paper treats the workshop only as evidence of the pipeline's robustness and reproducibility.

Two observations from the deployment bear directly on the pipeline. First, the tool proved operable by children with minimal instruction, which suggests that the constrained asset palette and the forgiving third-person navigation lowered the barrier to entry effectively. Second, the workshop build separated an authoring mode, in which assets are placed on the collision-enabled model, from a play mode, in which the space is explored; this separation was the friction participants noticed most, several of them expecting to build directly while moving through the space. That observation feeds the future work of Section 8.

7. Discussion

Three observations follow from the work. First, Blender is the indispensable bridge of the open-source pathway. In the absence of a Godot software development kit, it is Blender that hosts the conversion, repairs and optimises the geometry, and imposes the semantic structure the engine consumes; any comparable pipeline will need a component that plays this role. Second, collision geometry is the technical linchpin of integration. It is easy to overlook because it is invisible in the rendered scene, yet it is what makes both navigation and ray-cast placement possible, and its generation is the step most likely to fail silently on thin or complex geometry. Third, the combination of open data, open-source tools and CC0 assets removes the licensing and infrastructure barriers that usually accompany this kind of work, which matters most precisely in the schools and community centres where participatory planning takes place.

7.1 Comparison with the proprietary pathway

The proprietary route, using the PLATEAU software development kits for Unity or Unreal, offers a more direct import and a larger ecosystem of ready-made components. The open-source route described here trades that immediacy for three advantages that matter in the target setting: there are no licensing constraints on redistribution or classroom use, the runtime footprint is small enough for hardware without a dedicated graphics card, and the entire toolchain, together with the data and the asset library, can be shared and reproduced at no cost. The manual Blender stage is the price of this openness; automating it, as set out in Section 8, would narrow the convenience gap with the proprietary kits.

7.2 Transferability

Because the only assumption the pipeline makes about its input is that it is CityGML, the same four stages apply to any municipality in PLATEAU and, with adjustment of the coordinate-handling step, to CityGML datasets published elsewhere, including the national mapping agencies of Europe and the emerging OGC CityGML 3.0 ecosystem. What changes between contexts is the choice of source and target coordinate reference systems and the level of detail available, both of which are parameters of the conversion stage rather than structural features of the pipeline. A practitioner adopting the workflow therefore inherits the architecture and adjusts only the data-specific settings.

7.3 Limitations

Several limitations should be stated plainly. The conversion and optimisation stage still depends on manual work in Blender and has not yet been automated. Automatic collision generation requires human verification on thin or complex geometry. The selective level-of-detail strategy improves performance at the cost of some fidelity in the surrounding context. The quantitative performance characterisation defined in Section 5 is still in progress. Finally, the pipeline has so far been exercised on a single site and validated in a single workshop; broader testing across sites, hardware configurations and user groups is needed before its generality can be asserted.

https://doi.org/10.5194/isprs-archives-L-4-W1-2026-143-2026

8. Conclusion and future work

This paper has documented and validated a fully open-source geospatial pipeline that transforms CityGML data from Project PLATEAU into an interactive, real-time 3D environment running on low-specification hardware, using Blender and the Godot Engine. The pipeline fills the gap left by the absence of an official open-source software development kit, it proved robust in a demanding classroom deployment, and it is reproducible and transferable. Its principal weakness is also clear, and it sets the agenda for what follows.

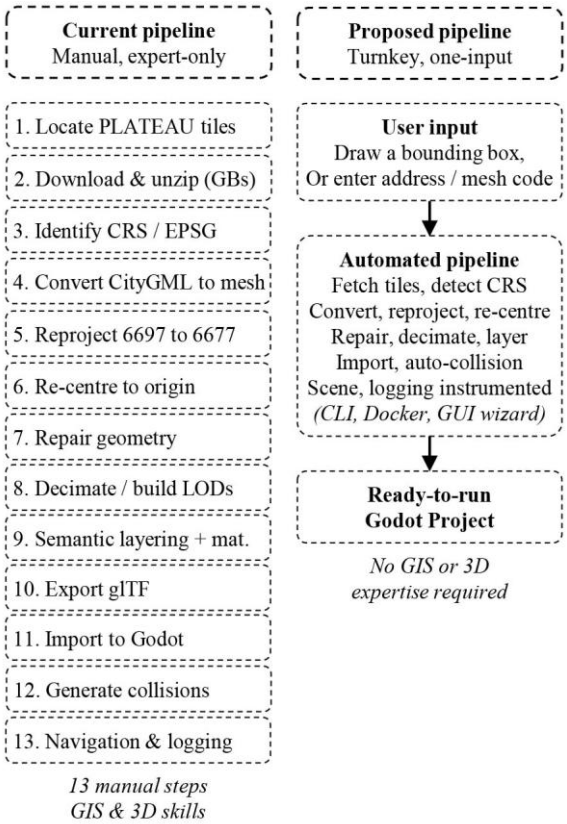

Figure 4. The current manual pipeline (left) requires on the order of a dozen tool-specific steps and GIS and 3D expertise; the proposed turnkey pipeline (right) collapses them behind a single user input.

8.1 Towards a turnkey, automated pipeline

The pipeline as described is reproducible, but it is not yet usable by a non-expert. As Figure 4 makes plain, reaching an interactive scene currently requires on the order of a dozen manual operations spread across a GIS, a 3D modelling environment and a game engine, from locating and downloading the correct mesh tiles, through reprojection, re-centring, geometry repair, decimation, semantic layering and export, to import, collision generation and the wiring of navigation and logging. Each step demands judgement and tool-specific knowledge, and an error early in the chain often surfaces only much later. A companion evaluation reached the same conclusion from the user side, noting that the data-preparation phase requires non-trivial GIS and 3D modelling expertise that limits adoption without dedicated facilitation (Martial et al., 2026b). The central direction of future work is therefore to collapse this manual chain into a turnkey procedure, so that an educator or a municipal officer can obtain a ready-to-run environment from a single, simple input.

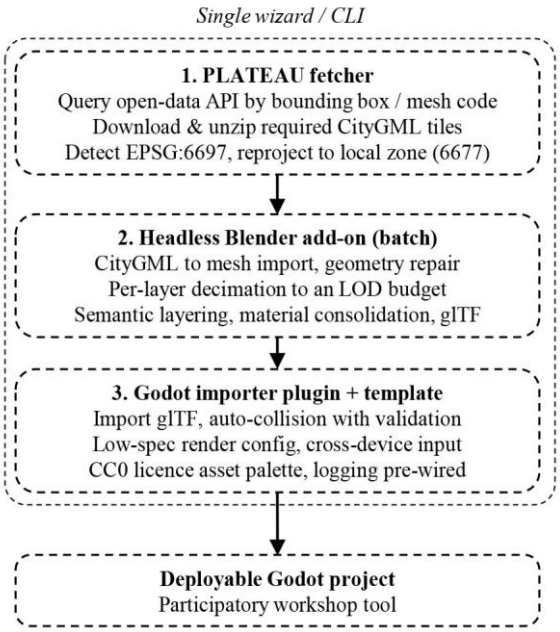

Figure 5. Proposed architecture of the automated toolchain: a PLATEAU fetcher, a headless Blender batch add-on and a Godot importer plugin with a template project, exposed behind a single wizard or command-line interface.

We propose a concrete architecture, shown in Figure 5, organised as three cooperating open-source components behind one interface. The first is a PLATEAU fetcher and converter: given a site specified as a bounding box drawn on a map, an address or a mesh code, it queries the open-data portal, downloads only the tiles that intersect the area of interest, detects the native EPSG:6697 frame and reprojects to the appropriate local metric zone, and converts CityGML to a mesh interchange format using the open converters discussed in Section 3.2. The second is a headless Blender add-on that runs the repair, the per-layer decimation against a stated LOD and polygon budget, the semantic layering and the material consolidation as a non-interactive batch, and exports glTF; running Blender headless turns the most expert-dependent stage into a repeatable, parameterised operation. The third is a Godot importer plugin shipped with a template project: it imports the glTF, generates collision geometry and, critically, validates it against the source mesh to flag the thin or complex cases that need attention, configures the low-specification render path and the cross-device input, and arrives with the CC0 asset palette and the logging routine already wired.

Several properties make this realistic rather than aspirational. Each component automates steps that are already performed manually and deterministically today, so automation is a matter of scripting existing operations rather than inventing new capability. The reprojection, tile selection and conversion rest on established open libraries; Blender exposes a complete Python interface that makes headless batch processing routine; and Godot's editor plugins and command-line export already support exactly the kind of automated scene assembly required. Packaging the whole as a command-line tool, a container image and a thin graphical wizard would let the same pipeline serve a developer, a continuous-integration job and a non-technical

workshop organiser without any of them touching the intermediate GIS steps. The one step that cannot be fully automated, collision validation on degenerate geometry, is precisely the one the importer should surface to the user rather than hide.

8.2 Further directions

Beyond automation, the pipeline should be benchmarked across different PLATEAU municipalities and LOD configurations to establish how its performance and the quality of automatic collision generation vary with urban form. The present deployment found its relevance at the macro scale of a large, open and largely flat site; extending the approach to the micro scale, encompassing building interiors, narrow streets and the finer treatment of street furniture, is the natural next frontier and will require both richer geometry and more careful collision handling. The authoring-and-play separation that participants noticed points towards an in-engine construction mode, in which the environment is built and modified within play and the resulting designs can be saved and shared, closing the gap between designing and exploring. Finally, because the entire toolchain is open, releasing it as a maintained template project with the automated components described above would let a community of practice form around the workflow, contributing assets, coordinate presets for additional municipalities and refinements to the collision and logging routines, so that the contribution shifts from a single documented pipeline to an openly governed resource.

Acknowledgements

The authors thank the staff of the participating elementary school, the Yokohama City Urban Development Bureau, and the university and graduate student facilitators for their support in conducting the workshop.

References

Adams, P.C., 1998. Teaching and learning with SimCity 2000. *Journal of Geography* 97(2), 47–55.

Ash, J., Gallacher, L.A., 2011. Cultural geography and videogames. *Geography Compass* 5(6), 351–368. doi.org/10.1111/j.1749-8198.2011.00427.x.

Holfeld, J., 2024. On the relevance of the Godot Engine in the indie game development industry. arXiv. doi.org/10.48550/arXiv.2401.01909.

Kenney, 2024. Kenney game assets (CC0). kenney.nl (9 June 2026).

Kolbe, T.H., Kutzner, T., Smyth, C.S., Nagel, C., Roensdorf, C., Heazel, C., 2021. *OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard*. OGC Document 20-010, version 3.0.0, Open Geospatial Consortium. docs.ogc.org/is/20-010/20-010.html (9 June 2026).

Kutzner, T., Chaturvedi, K., Kolbe, T.H., 2020. CityGML 3.0: new functions open up new applications. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 88(1), 43–61.

Levinzon, R., 2024. Godot Asset Placer (version 1.2.4). Godot Engine asset library and GitHub repository. github.com/levinzonr/godot-asset-placer (9 June 2026).

Martial, L., 2019. Une ville n'est pas un jeu. *Géographie et cultures* 109, 55–72. doi.org/10.4000/gc.9986.

Martial, L., Kubono, H., Miura, S., 2025. Game-based learning and machizukuri: gamification in youth participatory urban planning in Japan. Preprint, Chuo University.

Martial, L., Kubono, H., Miura, S., 2026a. Heatmap analysis of spatial design in a gamified machizukuri workshop using open-source tools. Forthcoming.

Martial, L., Kubono, H., Miura, S., 2026b. Evaluating a game-based workshop for youth urban design using open-source tools. Forthcoming.

Ministry of Land, Infrastructure, Transport and Tourism (MLIT), 2024. Project PLATEAU. www.mlit.go.jp/plateau/en (9 June 2026).

Project PLATEAU, 2024. PLATEAU GIS Converter. Project PLATEAU / MLIT, GitHub repository. github.com/Project-PLATEAU/PLATEAU-GIS-Converter (9 June 2026).

Seto, T., Furuhashi, T., Uchiyama, Y., 2023. Role of 3D city model data as open digital commons: a case study of openness in Japan's digital twin "Project PLATEAU". *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, XLVIII-4/W7-2023, 201–208. doi.org/10.5194/isprs-archives-XLVIII-4-W7-2023-201-2023.