Combining Open Source and Commercial Tools in Digital Twin for Cities Generation

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-11 (2022) 185–189

Combining Open Source and Commercial Tools in Digital Twin for Cities Generation

V. Naserentin[§, *,**] **S. Somanath**[*] **O. Eleftheriou**[*,***] **A. Logg**[*]

[*] *Chalmers University of Technology, Sweden*
[**] *Aristotle University of Thessaloniki, Greece*
[***] *University of the Aegean, Greece*
[****] *Corresponding author: vasilis.naserentin@chalmers.se*

Abstract: Evidently, Smart Cities are on the rise and there is an increasing need for digital twins of these complex environments and their corresponding 3D models. The creation and maintenance of such twins is a time consuming task, since cities are living evolving organisms. In this paper we are presenting work done within the *Digital Twin Cities Centre* (DTCC) in Sweden in the field of automated 3D city model generation. We showcase a novel method of combining open source and commercial software for creating digital twins of any urban context in a procedural way from raw input data and using Unreal Engine as a visualization front-end. We combine two different workflows, one based on the commercial software suite *Feature Manipulation Engine* and in parallel we utilize the open source code developed within the Centre, dubbed *DTCC Builder*. The assets created can be used in urban planning, multi-physics continuum mechanics simulations and for visualization and illustration on a 3D scale. By using, in a complementary way, established commercial software and state of the art open source C++ code we manage to utilize the quality of life features the first provides, while assigning the demanding tasks of comforming mesh generation to the former. The longer term goal is (nearly) real-time mesh digital twinning of any city and user interaction with the 3D assets provided.

Keywords: 3D city models, Digital Twins, FME, Unreal Engine, Data pipelines, Digital Prototyping

1. INTRODUCTION

The need for creating virtual copies of physical systems has been around for many years (Ketzler et al., 2020) and the term *twin* was introduced already by NASA's Apollo program decades ago (Liu et al., 2021)). Specifically, digital twins in the urban environment have been seeing increasing attention ((Ketzler et al., 2020)). On the city scale, the twin is required to serve a variety of purposes, including multi-physics simulations (Ranjbar et al., 2012; Rivas et al., 2019; Souhaib Douass and Kbir, 2020), what-if scenarios (Bahu et al., 2014) and life-cycle analysis (Strzalka et al., 2011) and thus the common ground for all the processes is a 3D city model. In the present work, we tackle the problem of creating *Level of Detail 1* (LOD1) (Biljecki et al., 2014) 3D models of any urban environment in the form of both surface and volume meshes. The mesh generation is based on publicly available datasets from the Swedish Mapping, Cadastral and Land Registration Authority (Lantmäteriet, the Swedish Mapping, Cadastral and Land Registration Authority, 2022) (LM) or any other similar repository that adheres to the data standards.

The presented mesh generation approach is innovative as it generates triangulated surfaces while also accommodating for *tetrahedral volume meshes*. The generated 3D model, provides a surface model along with a watertight high-quality computational volume mesh satisfying a certain minimal angle condition. These conditions have to be satisfied in order to use the model for high-performance simulations of, e.g. air quality computational fluid dynamics (CFD) and urban wind comfort. In a combination with geometries we also use additional types of data, such as road center-line data, land-use classification as well as forestry data from LM, in order to further enhance and enrich the visualization quality of the produced assets.

The ultimate goal is the creation of an *interactive* simulation environment which automates the computational mesh generation. To achieve an interactive process, the procedures involving the mesh generation must be *robust and efficient*, resulting in a mesh generation pipeline that never breaks down in case of unexpected or bad input data as well as ensuring minimal time in a simulation process. While these goals have been partially achieved by the presented software, ways for future optimizations that will try to approach real-time generation of 3D models and meshes are discussed.

⋆ This work is part of the Digital Twin Cities Centre supported by Sweden's Innovation Agency Vinnova under Grant No. 2019-421 00041, and GATE Project supported by the Horizon 2020 WIDESPREAD-2018-2020 TEAMING Phase 2 Programme under Grant Agreement No. 857155, the Swedish Research Council for Sustainable Development Formas (grants 2019-01169 and 2019-01885) and by Operational Programme Science and Education for Smart Growth under Grant Agreement No. BG05M2OP001-1.003-0002-C01. P.O. Hristov was funded by the National Scientific Program "Petar Beron i NIE" under the AUDiT project, no. KP-06-DB/3.

10.1016/j.ifacol.2022.08.070

2. DIGITAL TWIN CITIES CENTRE (DTCC) PLATFORM BUILDER

One of the most demanding and time consuming tasks in the urban scale simulation practise is the mesh creation and maintance (Stoter et al., 2020; Logg and Naserentin, 2022). The Builder (Logg and Naserentin, 2021) tries to tackle this by providing modeling and meshing capabilities for 3D city models of any urban context that has available cadastral building footprints and aerial point cloud (LiDAR). The code creates Digital Elevation and Surface Models (DEM and DSM respectively), volume and surface meshes as well as synthetic (randomised) cities. Some artifacts generated from Builder can be seen in Figure 2. The mesh generation process requires three steps and the first step is to generate a city model by combining 3D point cloud data with cadastral Geographic Information System (GIS) 2D data. The next step simplifies the model which was generated from the initial step. The last part of the process is using the resulted simplified model from the second step to generate the final mesh.

2.1 Step 1: Generate city model

To generate the initial city model we are using cadastral data to extract 2D building footprints. Then, in order to calculate building heights we are sampling the provided point cloud for points located inside the extracted footprint data. The result is a city model in LOD1 represented as polygonal prisms $P \times [h_0, h_1]$, where P is the 2D polygon representing the building, h_0 is the ground height at the centre of the building, and h_1 is the absolute computed height of the building.

2.2 Step 2: Simplify city model

The city model generated on the first step is in LOD1 which consists of a set of buildings. Each building is defined by 2D polygonal footprint data, ground height and roof height. To get any point within the computational domain we combine these data with a terrain map, allowing us to get the ground height. Before proceeding with the mesh generation, the need to simplify the city model arises as building footprints may introduce problems in cases where they overlap with each other or they are located at arbitrarily small distances. These problems can either make the mesh generation process extremely demanding by having to resolve tiny gaps between buildings or even break down the whole generation. Table 1 outlines the individual steps of the simplifications.

Step 2.1	Merge building footprints
Step 2.2	Clean building footprints
Step 2.3	Compute building heights
Step 2.4	Export city model

Table 1. Essential substeps for Step 2: City model simplification.

2.3 Step 3: Generate city mesh

The resulting city model from Step 2 is a simplified model which is ideal for the final mesh generation. The process starts off by generating a 2D mesh based on the building footprint data followed by layering and transforming the 2D mesh in order to generate a 3D volume mesh out of the city model. Table 2 outlines the key steps involved in the process. Figure 1 shows the output of each of the substeps of Step 3.

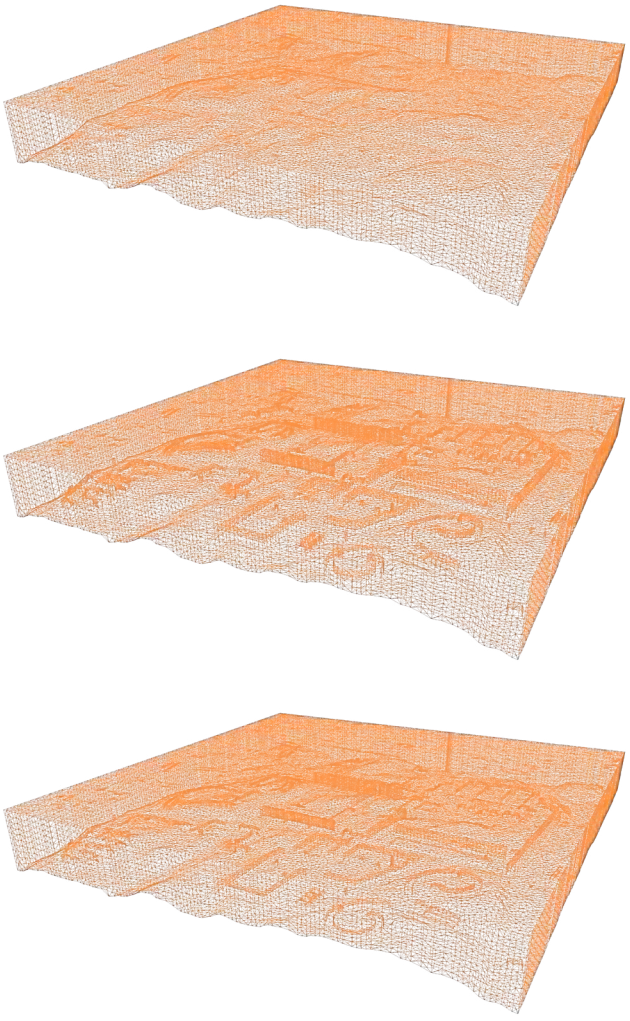

Fig. 1. Output from Steps 3.3–3.5 in the 3D mesh generation process for a dataset from Hammarkullen district in Gothenburg, Sweden. The figures show the boundaries of the generated tetrahedral volume meshes.

Step 3.1	Generate 2D mesh
Step 3.2	Generate 3D mesh (layer 2D mesh)
Step 3.3	Smooth 3D mesh (set ground height)
Step 3.4	Trim 3D mesh (remove building interiors)
Step 3.5	Smooth 3D mesh (set ground and building heights)
Step 3.6	Export mesh

Table 2. Essential substeps for Step 3: Mesh generation

The interior of buildings is found by first marking triangles in the ground mesh that are inside building footprints (using an efficient point-in-polygon test). During the trimming phase, the corresponding tetrahedra are removed that have been generated by layering such marked triangles, if also their midpoint is below the height of the building roof. The smoothing steps (3.3 and 3.4) are handled

by Laplacian smoothing; that is, by solving a PDE for the vertical displacements of the 3D mesh nodes in order to ensure that the mesh conforms to both the ground height and building heights.

The quality of the generated 3D mesh is ensured in three ways. First, the polygonal building footprints are preprocessed in order to ensure that each single polygon is of good quality (orientation, size, minimum clearance) and that the distance between any pair of polygons is larger than a given tolerance (otherwise the footprints are merged). Second, the underlying 2D mesh is generated using the Triangle mesh generator using a prescribed minimum quality for the generated mesh. Third, the 3D mesh generated by layering the 2D mesh is smoothed by Laplacian smoothing.

3. FME PIPELINE

Safe Software® Feature Manipulation Engine (FME) (Safe Software Inc., 2021) is a commercial data manipulation software that supports multiple file formats and predefined transformations through *transformers*. The advantage of using FME in the early stages of developing a data pipeline is the speed at which prototyping can be done. FME was used to develop and test pipelines for generating immersive visualisations in Unreal Engine (UE) (Epic Games, 2022) from raw GIS data in parallel to the DTCC Platform builder.

The FME pipeline (figure 3) begins with the Digital Terrain Model (DTM), building footprints, LiDAR point-cloud, road center-line, and land-use classification data and produces a series of raster tiles and a 3D building mesh that is eventually read by UE for the visualisation.

To create the UE raster tiles, first, a vector *mask* is generated for the various land-use classifications and roads. The boundaries between the vector masks are then interpolated to create a smooth transition between the layers. Finally, the resulting raster images are tiled and exported as images that can be read by UE.

The LiDAR point cloud, building footprints, and a DTM are used for the building meshes. Using the LiDAR point cloud, the median height of each building is first extracted and appended to the building footprint. The point clouds are not filtered or pre-processed. The centroid of each building footprint is then aligned to the ground using the DTM, and the footprints are extruded to their respective heights. The resulting buildings are then meshed and exported as an UE Datasmith file.

There are some drawbacks to this process. First, the predefined transformers that FME is built around do not expose all parameters that feed into the transformation. Though this expedites the iterative process of developing pipelines, it does not allow for the same flexibility as developing customised tools. Second, since FME is a commercial software, sharing the pipelines developed in FME is limited to those with commercial FME licenses.

4. FRONTEND IN UNREAL ENGINE

To visualize the data we have encoded them into grayscale images (data textures). By using grayscale images, we are clamping data values between range [0,1], thus allowing us to visualize them using any colormap of our choosing. In order to display both the data and the textures that were assigned to each mesh we have implemented a linear interpolation between them for each pixel. This technique allows us to simultaneously paint data values on top the assigned textures to maintain the "feel" of the mesh.

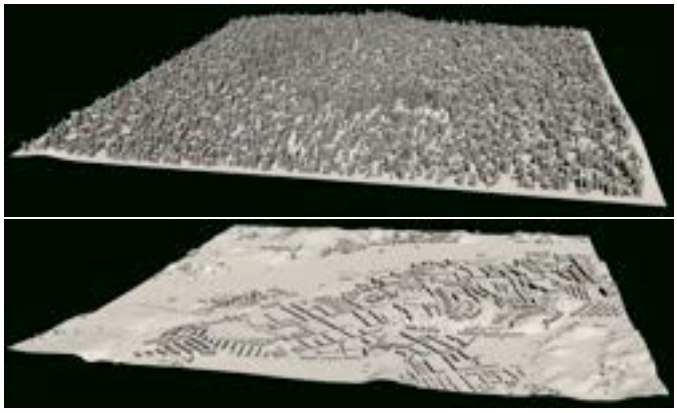

Fig. 2. Top: Surface mesh of a synthetic city randomly generated by Builder. Bottom: Surface mesh of a test dataset of Majorna district in Gothenburg, Sweden created by Builder.

By interpolating default textures with data values we have full control over the strength of the displayed values, meaning that we can control the opacity of both the default textures and the data values. Additionally, we scaled the data texture to match the whole area of interest and correlated the world location for each pixel in the engine with our data texture in order to drape data values on top of the corresponding coordinates.

The last step of this process was to pack the whole functionality in a material function that was used throughout all the materials in our meshes. This allowed the system to be used seamlessly over the entire project without manually modifying each material.

5. COMBINING THE APPROACHES

The UE pipeline uses data on the land use and vegetation as raster tiles and then overlays a 3D Building mesh as unreal Datasmith files (see figure 4). Generating data that can be read by UE is currently done using FME. However, there are benefits to combining the building mesh generation approaches.

Several discrepancies are visible when comparing the 3D buildings generated using the DTCC Builder pipeline and the FME pipeline. In the FME pipeline we do not filter anomalous points in the LiDAR data, resulting in incorrect buildings heights (see figure 5). The FME pipeline only produces a 3D building mesh. The terrain is unaffected. However, in the DTCC Builder pipeline there is specific care for the stitching of the building and ground meshes (see Section 2 and (Logg and Naserentin, 2022)) since the implementation takes care of the smoothing. This results in better vertical alignment between the building and the terrain when visualised.

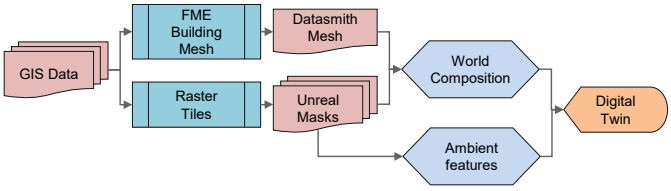

Fig. 3. FME building mesh generation workflow diagram.

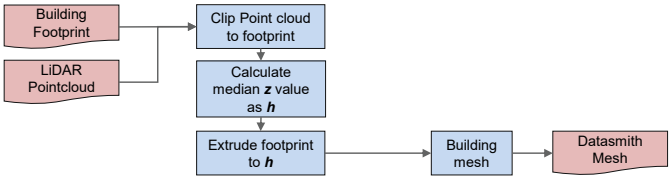

Fig. 4. FME workflow diagram to generate Unreal Engine tiles and Datasmith mesh.

While commercial software like FME is useful for prototyping new workflows, testing data pipelines and improving access to such pipelines, combining the two approaches produces the best possible result. The combined pipeline provides added visualization capabilities for the ambient features (including water bodies, green areas and road network) coming from the FME workflow, with the robust ground and building mesh generation of Builder. It has to be noted that currently, the two pipelines are combined offline, i.e. manually injecting the artefacts produced from Builder into the FME workflow. In following iterations, FME will be invoking the Builder internally, possibly through a container or via an API. Commercial tools such as FME can serve two roles: test the workflows before developing them into open-source tools like Builder. The second is to invoke Builder and consume the data produced, making the tools more accessible to users.

Additionally, the DTCC Builder generates 3D building meshes quicker than the FME pipeline. Using the FME pipeline, the UE assets are generated in 8 minutes, whereas the DTCC Builder pipeline requires around 5 minutes, running on the same computer hardware and the same raw data sources. Also the FME pipeline automatically performs optimisations through load balancing and parallelization, whereas the DTCC Builder is executed on a single thread.

Figure 6 shows the final 3D visualisation in UE. 3D buildings and terrain generated using the DTCC Builder pipeline are used.

6. DISCUSSION & CONCLUSIONS

Visualisation of 3D data is an essential component of a digital twin. The quality of the 3D models generated and the data processing pipelines' accuracy, convenience, and speed are important steps in visualising these digital replicas. The pipelines developed using commercial tools are useful for quick prototyping. However, the workflows developed using the DTCC Builder pipeline produces watertight volume meshes that are accurate to the real-world data and take into account anomalies in the raw data. Additionally, they offer a higher degree of control on the mesh generation (Logg and Naserentin, 2022). Our workflow has the limitation that it offers only LOD1,

Fig. 5. Mesh comparison between results of FME and DTCC Builder workflows (Red - FME, Blue - DTCC Builder)

Fig. 6. 3D visualisation of test area in Unreal Engine.

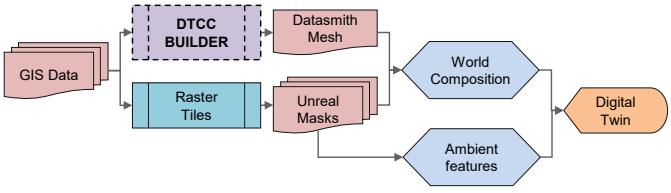

Fig. 7. Combining DTCC Builder workflow with the FME workflow

but there is significant research towards in using Machine Learning for adding non-flat rooftops in next prototype iterations. Moreover, DTCC Builder offers no parallelism, but it is planned to add shared memory architecture capabilities, since the longer term goal would be (near) real-time mesh creation and manipulation.

There are also some other lesser limitations that appear due to mixing the two pipelines. Combining workflows built on different platforms can lead to interoperability issues in handing over the data. Parsing the data produced by DTCC Builder adds additional overhead to the workflow, and can be time consuming in comparison to reading raw GIS data. Future development will address these issues to further improve the efficiency of the workflow and ensure that gains from one pipeline are not lost to another.

A combination of the two approaches is presented in this paper to produce front end visualisations in UE. The workflow combines the benefits of rapid prototyping using FME and robust mesh generation pipelines using DTCC Builder to further develop robust, efficient and open-source pipelines.

7. DATA AVAILABILITY

Our implementation, as well as data needed to reproduce the results presented, are available as free open-source data (under the permissive MIT license) Logg and Naserentin (2021). Regarding the FME workflow the lead author can provide details about the exact implementation process.

REFERENCES

Bahu, J.M., Koch, A., Kremers, E., and Murshed, S.M. (2014). Towards a 3d spatial urban energy modelling approach. *International Journal of 3-D Information Modeling (IJ3DIM)*, 3(3), 1–16.

Biljecki, F., Ledoux, H., Stoter, J., and Zhao, J. (2014). Formalisation of the level of detail in 3D city modelling. *Computers, Environment and Urban Systems*, 48, 1–15.

Epic Games (2022). Unreal Engine. URL `https://www.unrealengine.com`.

Ketzler, B., Naserentin, V., Latino, F., Zangelidis, C., Thuvander, L., and Logg, A. (2020). Digital Twins for Cities: A State of the Art Review. *Built Environment*, 46(4), 547–573. doi:10.2148/benv.46.4.547.

Lantmäteriet, the Swedish Mapping, Cadastral and Land Registration Authority (2022). URL `https://www.lantmateriet.se/en/`.

Liu, M., Fang, S., Dong, H., and Xu, C. (2021). Review of digital twin about concepts, technologies, and industrial applications. *Journal of Manufacturing Systems*, 58, 346–361. doi:10.1016/J.JMSY.2020.06.017.

Logg, A. and Naserentin, V. (2021). Digital Twin Cities Platform — Builder. URL `https://gitlab.com/dtcc-platform/dtcc-builder`.

Logg, A. and Naserentin, V. (2022). Digital twins for cities: Automatic, efficient, and robust mesh generation for large-scale city modeling and simulation. Submitted in *Nature Computational Science*.

Ranjbar, H.R., Gharagozlou, A.R., and Nejad, A.R.V. (2012). 3d analysis and investigation of traffic noise impact from hemmat highway located in tehran on buildings and surrounding areas. *Journal of Geographic Information System*, 04, 322–334. doi: 10.4236/jgis.2012.44037.

Rivas, E., Santiago, J.L., Lechón, Y., Martín, F., Ariño, A., Pons, J.J., and Santamaría, J.M. (2019). Cfd modelling of air quality in pamplona city (spain): Assessment, stations spatial representativeness and health impacts valuation. *Science of The Total Environment*, 649, 1362–1380. doi:10.1016/J.SCITOTENV.2018.08.315.

Safe Software Inc. (2021). Feature Manipulation Engine (FME). URL `http://safe.com`.

Souhaib Douass, M. and Kbir, A. (2020). Flood zones detection using a runoff model built on hexagonal shape based cellular automata. *International Journal of Engineering Trends and Technology*, 68. URL `http://www.ijettjournal.org`.

Stoter, J., Ohori, A., Dukai, B., Labetski, A., , Kumar, K., Vitalis, S., and Ledoux, H. (2020). State of the art in 3d city modelling. URL `https://www.gim-international.com/content/article/state-of-the-art-in-3d-city-modelling-2`.

Strzalka, A., Bogdahn, J., Coors, V., and Eicker, U. (2011). 3d city modeling for urban scale heating energy demand forecasting. *HVAC and R Research*, 17, 526–539. doi: 10.1080/10789669.2011.582920.