







On Provisioning Procedural Geometry Workloads on Edge Architectures

Iliir Murturi¹^a, Chao Jia²^b, Bernhard Kerbl²^c, Michael Wimmer²^d, Schahram Dustdar¹^e
and Christos Tsigkanos¹^f

¹*Distributed Systems Group, TU Wien, Vienna, Austria*

²*Research Unit of Computer Graphics, TU Wien, Vienna, Austria*

Keywords: Edge Architectures, Computational Workloads, Edge-cloud Continuum.


Abstract: Contemporary applications such as those within Augmented or Virtual Reality (AR/VR) pose challenges for software architectures supporting them, which have to adhere to stringent latency, data transmission, and performance requirements. This manifests in processing 3D models, whose 3D contents are increasingly generated procedurally rather than explicitly, resulting in computational workloads (i.e., perceived as Procedural Geometry Workloads) with particular characteristics and resource requirements. Traditionally, executing such workloads takes place in resource-rich environments such as the cloud. However, the massive amount of data transfer, heterogeneous devices, and networks involved affect latency, which in turn causes low-quality visualization in user-facing applications (e.g., AR/VR). To overcome such challenges, processing elements available close to end users can be leveraged to generate 3D models instead, and as such the edge emerges as a central architectural entity. This paper describes such procedural geometry workloads, their particular characteristics, and challenges to execute them on heterogeneous devices. Furthermore, we propose an architecture capable of provisioning procedural geometry workloads in edge scenarios.


1 INTRODUCTION


Contemporary applications such as those within Augmented or Virtual Reality (AR/VR) demand dedicated software architectures, able to cope with stringent latency, data transmission, and performance requirements. Those challenge the traditional cloud-centric view, where computation and data reside in powerful cloud servers, but away from user-facing applications which may have to further overlay sensory information obtained near user's locations. The key computational functionality within AR/VR consists of processing high-quality 3D models, whose 3D contents are increasingly generated procedurally rather than explicitly, resulting in computational workloads with particular characteristics and resource requirements. Visualizing and representing 3D models is a resource-intensive process that involves both massive data transfer to user clients (in terms of both volume and velocity), as well as demanding computation, with latency perceived by end users being a


major concern. In recent years, one prominent approach that has emerged to overcome latency delays especially in pervasive applications suggests utilizing computation entities in proximity to end users. Edge Computing is a distributed computing paradigm that places resources (e.g., compute and storage) at the edge of the network (Shi and Dustdar, 2016). Edge resources (i.e., perceived as *edge devices*) are typically resource-constrained and heterogeneous devices that can process, store, and analyze data and deliver efficient and low-latency user-facing services. Such edge devices can support applications by i) running decision functions close to data-producing end users, ii) processing computational workloads, and iii) minimizing the need to transmit data to the cloud.


Accurate digital architectural models are essential to various practical applications such as urban planning (Vanegas et al., 2009), 3D navigation (Hildebrandt and Timm, 2014), natural and social phenomena simulation (Heuveline et al., 2011; Jund et al., 2012) in urban environments. Employing such 3D representations in contemporary near real-time applications requires their abstraction or simplification (Visconti et al., 2021) to avoid storage and transmission of the sheer volume of geometric information that they include. One way to improve the user-perceived quality of these visual models without mas-


^a <https://orcid.org/0000-0003-0240-3834>

^b <https://orcid.org/0000-0003-2304-5976>

^c <https://orcid.org/0000-0002-5168-8648>

^d <https://orcid.org/0000-0002-9370-2663>

^e <https://orcid.org/0000-0001-6872-8821>

^f <https://orcid.org/0000-0002-9493-3404>

sive storage overhead is to procedurally generate the geometric data on the fly. Procedural modeling has been well studied in the research area of computer graphics and successfully applied in video games, the movie industry, and AR/VR applications where compelling and immersive virtual urban environments are vital to the visual experience (Kim et al., 2018; Müller et al., 2006). By recursively applying a set of rules that represent spatial transformations on geometric shapes, procedural modeling can generate highly detailed geometry starting from a very basic primitive shape called an axiom (e.g., a box or a quadrilateral). For example, windows or doors on a wall can be created by subdividing a rectangle and applying extrusion operations and different textures to the resulting small rectangles. Analogously, details on windows or doors can be generated with further subdivision, extrusion, and texture mapping rules.

The expressiveness of procedural geometry allows for a compact representation of complex 3D physical environments with minimal resources. Moreover, procedural geometry is highly flexible in that the generation of geometric data can be tailored to different requirements by controlling the set of rules to be evaluated. Recently, efforts have also been made to harness advances in parallel computing to significantly speed up procedural geometry generation (Steinberger et al., 2014). These particular characteristics enable the distribution of procedural geometry workloads on a wide range of devices to raise performance. Aside from data center-grade servers, edge devices such as single-board computers (SBC) equipped with low-cost processors and embedded systems specialized in parallel processing can also be leveraged for load balancing and latency reduction.

Edge-based infrastructures are heterogeneous and dynamic environments consisting of various devices featuring different capabilities; resources available may differ in terms of computational capabilities (i.e., from low-powered devices to server-grade hosts). Such heterogeneity poses several challenges from a software architecture perspective when executing procedural geometry workloads since they have particular processing requirements and low-latency demands. The key theme is that processing elements close to end users can be leveraged to generate 3D models, to avoid user-perceived latency. In this paper, we identify the current trend towards distributed computing for visualization tasks, describe such procedural geometry workloads, their particular characteristics, and challenges to execute them. Furthermore, we propose an architecture capable of provisioning procedural geometry workloads in edge settings. Finally, we outline a research agenda.

2 PERVASIVE VISUALIZATION

With the arrival of ubiquitous network connectivity and affordable, consumer-grade smartphones and tablets, 2D and 3D visualizations that depend on live remote data (e.g., video streaming, games and simulations) have become available on mobile devices. In many cases, the raw data that must be transferred in real-time is either limited or can be sufficiently compressed, and end-user devices are capable of decompressing and performing the necessary visualization tasks themselves. However, the concept of the thin client is progressively becoming more important for high-end pervasive visualization: complex visual applications (e.g., Triple-A games) increasingly rely on cutting-edge or specialized hardware capabilities, opening the door for cloud gaming services that relieve end-user devices of these requirements. The established capabilities for streaming video at high resolutions can be exploited to deliver visual content that is entirely generated by dedicated cloud services, such as NVIDIA's GeForce Now, Microsoft's XCloud or Google's Stadia. However, as has become evident during the COVID-19 pandemic, constant streaming of high-resolution image data takes a heavy toll on the available internet infrastructure. With the imminent availability of yet faster 5G connectivity (high bandwidth, low latency, low range) in many areas, the exploitation of edge nodes is a logical next step to both improve user experience and relieve some of the infrastructure stress. However, edge computing is still a relatively young concept and in the process of being developed. In contrast to cloud services, performing visualization tasks with the help of edge nodes enables a range of scheduling and distribution strategies and encourages the development of specialized solutions for different use cases. Reasonable approaches that economize on the available resources require careful analysis of the implied workloads and the design of multi-tiered, dynamic architectures.

3 PROCEDURAL GEOMETRY AS A COMPUTATIONAL WORKLOAD

Online repositories such as Google Earth¹ or 3DCityDB² provide 3D models for several metropolitan cities. Those are particularly useful e.g., in navigation applications that aim to assist end-users while roaming inside a city. Unfortunately, the 3D representa-

¹Google Earth, <https://www.google.com/earth>

²The CityGML Database, <https://www.3dcitydb.org/>

tions of those architectural models used in everyday applications are often oversimplified and lack in terms of quality when requested in high-quality 3D due to the high latency caused by network congestion. As a result, applying the aforementioned methodologies to applications like AR/VR may result in unexpected user-perceived latencies. Thus, novel applications in AR/VR require novel paradigms and software architectures to cope with demands on latency, resource management, and computation requirements.

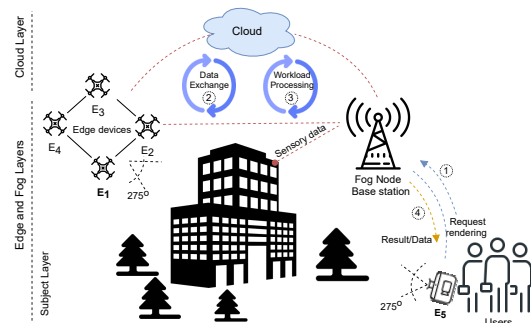


Figure 1: An example of an AR application reliant on high-quality 3D data for use in civil engineering.

AR/VR applications are increasingly being pursued in a plethora of fields ranging from gaming and immersive tourism to industrial applications. Consider for instance an AR application within civil engineering (Figure 1), which, using CityGML models, enables visualization of buildings by rendering them into users' smart devices such as smartphones, tablets, or 3D glasses. Use cases may involve users such as city inspectors roaming in the analyzing energy infrastructure (i.e., heating or cooling) (Kaden and Kolbe, 2014). In such a case, 3D visualization combined with real-time sensor data (e.g., energy consumption) enables users to perceive energy demands for each building on their smart devices.

However, to enable the interaction with high-quality 3D visualization on the users' smart devices, one must update the geometry of the observed building(s) as the user moves. Procedural geometry generation is an intensive process, while representing high-quality 3D visualizations on the user's smart device at interactive rates demands fast processing and low latency. To meet these demands, a user's smart device should take advantage of available computing resources in proximity (i.e., server-grade fog devices) to find the most suitable place to execute geometry workloads (1); this may take place in single-board computers, in server-grade fog nodes in mobile base stations, or in the cloud. After a computation request is made, various data is exchanged between compu-

tation entities and system components (2-3) to decide where to compute the workload as well as to provide the resulting geometry to the user's device (at the subject layer).

The ever-growing demand for high fidelity graphics and visualization applications entails processing enormous geometry data consisting of numerous triangles; for instance, around 1 million triangles are required to model an area of 0.05 km^2 with a moderate level of detail (LOD) near the center of the city of Vienna, resulting in at least 30 MiB of raw mesh data. In contrast, an analogous procedural generation of raw mesh data requires less than 0.1 MiB in input parameters that define the position, dimension and orientation of each axiom shape, as well as style parameters upon which detailed geometries can be generated. These style input parameters can for instance be extracted from high-quality imagery (aerial or drive-through) and applied on top of CityGML models in which buildings are represented by simple boxes that serve as a geometric baseline for the procedural generation (Figure 2).

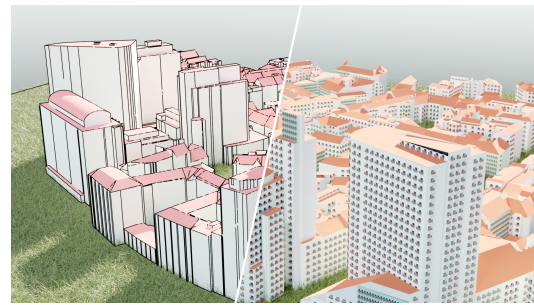


Figure 2: Illustration of procedural generation workload. Starting from a CityGML baseline model (left), detailed geometry is generated and delivered to the end user (right).

Given the amount of required raw mesh data, procedural generation workloads typically involve very intensive computation. This manifests as a trade-off for the full 3D data that would be equivalently transferred. The automatic generation of 3D data is configurable; one may compute a certain level of detail only, do so dynamically and depending on computation or time budgets. The (configurable) level of detail emerges naturally as another factor subject to optimization. Moreover, since geometric details for areas far away from the current point of view of the user are insignificant, their computation can be omitted, and rule evaluation can be terminated at a lower LOD for those buildings, thereby greatly eliminating unnecessary computation. Finally, the generated mesh data for areas that are frequently requested by end users can be cached and reused.

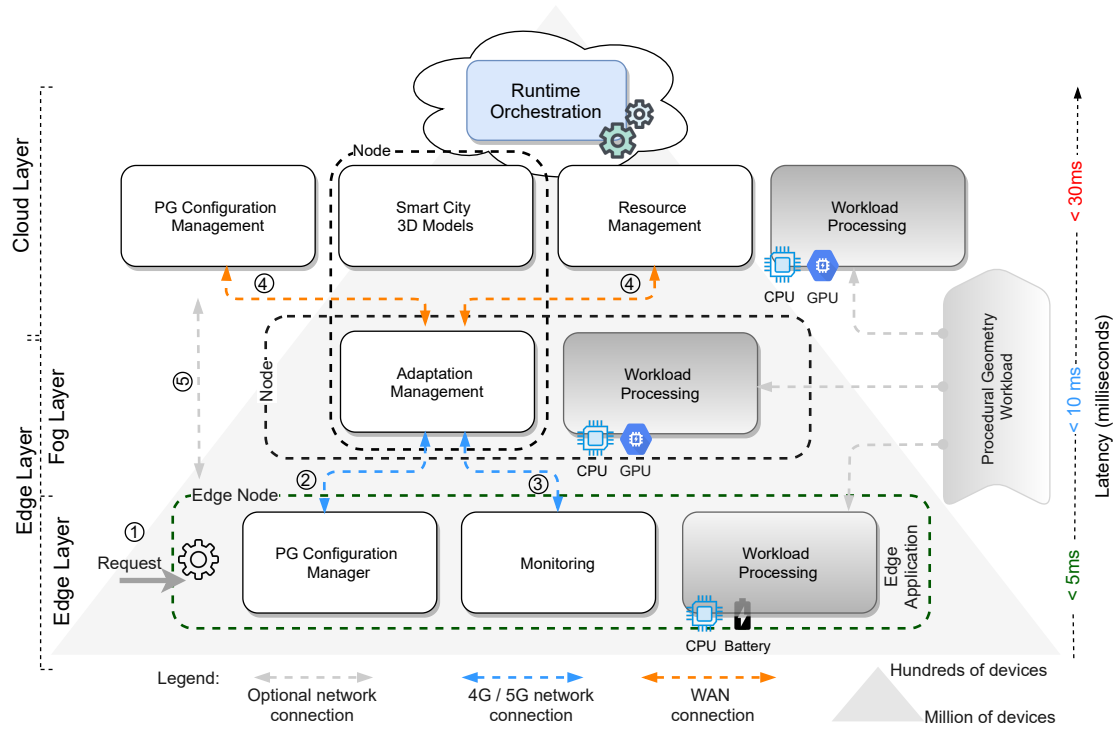


Figure 3: An overview of the proposed architecture and the interactions between layers.

4 SOFTWARE ARCHITECTURE CONSIDERATIONS

Edge Computing is positioned as an important architectural layer between cloud and end users (Figure 1). A platform for processing 3D models may leverage the decentralized nature of such infrastructures, consisting largely of three entities: first, the user that requests 3D processing; second, the software components that support finding the most suitable entity to execute the procedural geometry workload and cope with the dynamicity and uncertainty of the edge environment; and third, an orchestration layer, typically located in the cloud and responsible for monitoring the overall system and deployment of software components in the Edge-Cloud infrastructure.

The cloud is a resource-rich environment and provides advanced features for both service providers and service consumers; thus, we advocate its role for orchestration, performing 3D model generation, and providing the required resource management. The edge layer comprises devices placed in proximity to end users with different characteristics and hardware configurations, divided into sub-layers of i) fog

(i.e., powerful devices) and ii) edge (low-powered devices).

The fog layer includes a set of stationary powerful devices (i.e., physical or virtual) with different hardware configurations (i.e., CPU, GPU, storage, etc.). The role of the fog layer is interchangeable from executing geometry workloads with low latency to an intermediary layer for managing, communicating, and exchanging resources between different edge devices and the cloud. In addition, fog devices may provide storage for 3D models with different sizes (i.e., from MB to GB). The edge layer consists of numerous devices that can simultaneously request visualization; those are usually resource-constrained devices (i.e., CPU-based and battery-powered) — however, often with enough hardware capabilities to execute specific geometry workloads (such as modern smartphones).

Figure 3 illustrates different architectural manifestations to execute procedural geometry workloads. Edge applications operated by end users and hosted on smart devices interact with other system components deployed in the Edge-Cloud continuum. An edge application internal view consists of three components: i) procedural geometry configuration manager, ii) monitoring, and iii) workload processing.

The configuration manager enables users to express their goals (e.g., visualizing based on the user's location and within a specified radius). Essentially, a user configures spatial boundaries (i.e., a building or a neighborhood), and the amount of detail desired to represent the virtual world. The monitoring component is responsible for observing internal hardware resource states: i) internal hardware resources and their utilization and ii) quality of the communication link (i.e., status, latency, and bandwidth). The workload processing component is responsible for executing procedural geometry workloads (i.e., workloads can be packaged into software containers within the overall service-based architecture (Dustdar and Murturi, 2021)). As illustrated in Figure 3, the process starts (1) when a user expresses her goal via the edge application. Then, the request with hardware information is forwarded (2-3) to the adaptation manager which interprets the goal and decides where to execute the workload.

The adaptation management component is responsible for identifying devices needed to achieve a user goal with the lowest possible latency. If the goal is achievable on the user's device, it forwards the required data to the host. If the goal is not achievable locally, the adaptation component attempts to generate a deployment plan that maps the workload to other available devices. As illustrated in Figure 3, procedural generation may occur in the cloud, fog, and edge devices. To generate valid deployment plans, adaptation must consider several factors such as device hardware requirements, network metrics, and the time required to transfer (un)processed geometry. To generate optimal plans, the adaptation component requires fine-grained information of the infrastructure (4).

The cloud part has a supportive role, which includes procedural geometry configuration management, data storage (e.g., 3D models), resource management, geometry workload generation, and overall orchestration. As illustrated in Figure 3, the user's request can be forwarded (5) directly to the cloud as well if no other solution is feasible. The resource management component comprises a set of functionalities from resource discovery (i.e., discovering available edge devices) to context monitoring (i.e., monitoring hardware infrastructure and updating its status when changes occur). Orchestration entails where the software components must be placed, aiming for reliable and low-latency service to end users.

Recent developments in IoT-based systems have shown that systems can be engineered, deployed, and executed in Edge-Cloud infrastructures (Alkhabbas et al., 2020). At the same time, software components can easily self-adapt to dynamic changes in their de-

ployment topologies when the quality of their services is degraded (Brogi et al., 2020). Finally, as shown in Figure 3, software components can be placed on different devices yielding different deployment configurations. More specifically, software components that face high requests from a particular region can be placed in proximity to the end-users. For instance, if the procedural geometry generation for a particular city area occurs mostly on the user devices, then the orchestration mechanism must instantiate the data storage component with associated data (i.e., 3D models) on the nearest fog devices to the users. As a result, data can be forwarded faster to the end-users from the edge layer rather than from the cloud via WAN connection.

5 AN EMERGING RESEARCH AGENDA

Satisfying the dynamic and stringent requirements of contemporary applications such as those in AR/VR is challenging for centralized cloud-based systems. Processing 3D models and transferring vast amounts of data to user-facing devices over the internet incur latencies and result in user experience degradation. We discussed aspects emerging from latency and computation requirements and how edge architectures can address the requirements and support procedural geometry workloads. Thus, we sketched an architecture capable of provisioning such workloads in edge computing scenarios.

As future work, we aim at providing a complete technical framework for the processing of geometry workloads on edge-based architectures; this includes both technical and architectural aspects. Encapsulating procedural generation appropriately such as it being able to execute on heterogeneous hardware platforms is a challenging task, as such workloads are required to take advantage of specialized hardware (such as GPUs) when available, yielding different configurations. Subsequently, our vision entails them to be containerized, such that a service-based architecture emerges across the device-to-cloud continuum. Performance aspects of different geometry workloads executed on state-of-the-art resource-constrained and powerful devices need to be carefully considered. Besides that, assessing deployment tradeoffs in terms of quality, performance, and cost is highly desired. Regarding deployment, the edge topology may not be static, and components may need to be scaled or migrated to comply with other constraints like energy, latency, or device movement, introducing dynamicity. Finally, we identify three main

research challenges that must be further investigated in the future:

- **Procedural Geometry Workload Configuration.** A platform for the described scenarios needs to hide operational complexity from application end users and developers. In particular, developers should be able to express in a high-level way the context in which particular procedural geometry executions are allowed to run. Thus, a novel domain-specific language (DSL) for specifying the high-level constraints such as Quality of Service (QoS) and hardware requirements remains a critical task.
- **Resource Discovery across the Edge-cloud Continuum.** A fundamental aspect in described scenarios is discovering resources such as edge devices or IoT resources (e.g., sensors, actuators, etc.) as they become available in the city's environment (Murturi and Dustdar, 2021). Thus, resource discovery in heterogeneous and dynamic edge-based settings remain among the main research challenges.
- **Resilient Geometry Workload Runtime.** The end users and edge devices providing computational resources can be mostly mobile. As a result, preserving optimal QoS in the face of client or resource mobility is another prominent research challenge that needs to be addressed in the future.

ACKNOWLEDGMENT

Research supported in part by the Research Cluster “Smart Communities and Technologies (Smart CT)” at TU Wien, the EU’s Horizon 2020 Research and Innovation Programme under grant agreement No. 871525 and by Austrian Science Foundation’s (FWF) project M 2778-N “EDENSPACE”.

REFERENCES

- Alkhabbas, F., Murturi, I., Spalazzese, R., Davidsson, P., and Dustdar, S. (2020). A goal-driven approach for deploying self-adaptive iot systems. In *2020 IEEE International Conference on Software Architecture (ICSA)*, pages 146–156. IEEE.
- Brogi, A., Forti, S., Guerrero, C., and Lera, I. (2020). How to place your apps in the fog: State of the art and open challenges. *Software: Practice and Experience*, 50(5):719–740.
- Dustdar, S. and Murturi, I. (2021). *Towards IoT Processes on the Edge*, pages 167–178. Springer International Publishing, Cham.
- Heuveline, V., Ritterbusch, S., and Ronnas, S. (2011). Augmented reality for urban simulation visualization. *Preprint Series of the Engineering Mathematics and Computing Lab*, (16).
- Hildebrandt, D. and Timm, R. (2014). An assisting, constrained 3d navigation technique for multiscale virtual 3d city models. *GeoInformatica*, 18(3):537–567.
- Jund, T., Kraemer, P., and Cazier, D. (2012). A unified structure for crowd simulation. *Comput. Animat. Virtual Worlds*, 23(3-4):311–320.
- Kaden, R. and Kolbe, T. H. (2014). Simulation-based total energy demand estimation of buildings using semantic 3d city models. *International Journal of 3-D Information Modeling (IJ3DIM)*, 3(2):35–53.
- Kim, J., Kavak, H., and Crooks, A. (2018). Procedural city generation beyond game development. *ACM SIGSPATIAL Special*, 10(2):34–41.
- Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Gool, L. V. (2006). Procedural modeling of buildings. *ACM Trans. Graph.*, 25(3):614–623.
- Murturi, I. and Dustdar, S. (2021). A decentralized approach for resource discovery using metadata replication in edge networks. *IEEE Transactions on Services Computing*.
- Shi, W. and Dustdar, S. (2016). The promise of edge computing. *Computer*, 49(5):78–81.
- Steinberger, M., Kenzel, M., Kainz, B., Müller, J., Wonka, P., and Schmalstieg, D. (2014). Parallel generation of architecture on the GPU. *Comput. Graph. Forum*, 33(2):73–82.
- Vanegas, C. A., Aliaga, D. G., Benes, B., and Waddell, P. (2009). Interactive design of urban spaces using geometrical and behavioral modeling. *ACM Trans. Graph.*, 28(5):111.
- Visconti, E., Tsigkanos, C., Hu, Z., and Ghezzi, C. (2021). Model-driven engineering city spaces via bidirectional model transformations. *Software and Systems Modeling*, pages 1–20.