Abstract—At the present, we are in the midst of another major societal shift that is leading to a hyper-connected society. In this hyper-connected society, the convergence of mobile technology, online access and global media is connecting masses of people without precedents, with resulting impacts on communication, content creation and social engagement. The deployment of interactive mobile services is, therefore, a key strategy for entities that seek to increase user engagement and awareness. However, the heterogeneity of the existent data and the limited hardware capabilities of mobile devices have been a technological challenge for designers of such interactive tools. In this paper, we introduce an interoperable framework for the visualisation and processing of 3D city models on mobile devices. Our solution crystallises on CityGML’s ability to store and describe 3D city-related geometry along with its semantics, which is fundamental to city level operations like thematic queries, analysis tasks, automatic integration, validity checking, and spatial data mining. The streaming of 3D models is transparent to the user, who perceives remote models as locally stored. The user interface is designed for 3D data exploration and analysis on mobile environments, and it is fully compliant with OGC open web standards.

I. INTRODUCTION

The concept of using geographical information systems (GIS) to analyse and support geospatial processes such as disaster management [16], urban planning [17], mobility [2], and management of environmental resources [4] is not completely new. Yet, for entities that integrate collaborative processes, the biggest challenges remain: 1) the lack of a common representation that goes beyond visual and structural characteristics of the data; and 2) the implementation and adoption of a modelling approach that supports integration, storage, exchange and visualisation of geospatial information, as well as operations like validity checking, automatic integration, thematic queries, and spatial data mining.

CityGML [10] is an open data model standard for the representation and exchange of 3D city models (vegetation and roads, buildings, terrain elevation, among others). It was designed to facilitate and support processes that require the interoperability and the use of the same.

Unlike desktop solutions, where the visualisation and navigation of CityGML models has been around for a while, the visualisation of CityGML on mobile devices is still an open challenge. Right now, the technological limitation is that Desktop solutions are upgradeable with memory and processing capabilities so they can handle the large size models, but mobile devices are not so flexible. In addition, the complexity of implementing a mobile solution increases with the variety of data that has to be visualised, mostly because of small displays, which require dedicated interfaces, and the mobile environment itself. For example, its use when walking or driving is a challenge to traditional ways of information presentation and user interaction.

In this paper, we describe an interoperable mobile framework for the navigation, visualisation and processing of large 3D city models. In section 2, we survey the relevant literature, both to see if similar studies have been done, and to define the framework from which to evaluate the relevance and impact of this study. The third and fourth section describe the proposed methodology as well as its implementation details. Lastly, section 5 wraps up possible extensibility.

II. RELATED WORK

There are a few solutions that aim at visualising the geometry and semantics of CityGML data models. Mao [8] suggested the conversion of CityGML geometries to more efficient formats like X3D, which are then visualisable with frameworks like X3DOM [3]. However, the exportation procedure does not preserve the semantic information of the model. Moreover, its use in interactive systems either requires pre-processing or, as suggested by Over et. al. [12], the usage of bounding boxes to delimit the area of interest.

The visualisation of pre-generated images on a web browser from pre-defined viewpoints, using a Web Perspective View Service (WPVS) was proposed by [18]. The main drawback of this solution is that it delivers images instead of 3D models. Hence, 3D navigation of the model is limited because of its fixed perspective. Further limitations include the lack of support for information retrieval or spatial analysis, which were conceptually addressed with the introduction of the Web View Service [13] and Web 3D Service [1]. The Web 3D Service (W3DS) delivers optimised scene graphs to the client application, whereas the Web View Service (WVS) delivers ready for display projected images that correspond to the requested view. Complementary, the Symbology Encoding (SE) and Styled Layer Descriptor (SLD) [11] standards enable user-defined styling definitions. A similar approach to W3DS was proposed by Prandi et al. [13] for CityGML.

Kumke et al. [7] described a desktop tool for the visualisation of CityGML data models but it was not optimised for visualisation of large 3D models. Rakkolainen et al. [14] and...
Nurminen et al. [9] proposed two mobile applications for the visualisation of 3D cities. However, neither the applications are interoperable nor they provide support for CityGML's semantic capabilities.

This paper proposes a new methodology for modelling city data, from the generation of 3D buildings to its visualisation on mobile devices. The solution is driven by the use of CityGML as the reference data model. We adopted CityGML not only because it has been accepted as an OGC standard, but mostly because it supports the representation of sets of 3D urban objects (terrain elevation, buildings, land use, vegetation and roads, among others) in a single data model, thus facilitating the use and the interoperability of the same. Our solution combines the flexibility of using standard Web services for the processing and analysis of non-geometrical information with the efficiency of presenting geometric information locally.

III. Methodology

The section describes the steps required to transform raw data into information: generation, storage, access, processing and visualisation.

Our first step was to use novaFACTORY software to automatically generate 3D CityGML models with roof structures from building footprints, LIDAR point clouds and digital terrain models. The CityGML data model, which was stored in a CityGML database, was then augmented with data retrieved from sources such as Open Street Maps, proprietary orthophotos, and statistical data (e.g. energy consumption, noise, etc) and exposed using standard web services to ensure the interoperability and integrity of the dataset. A mobile application was finally used to explore and visualise the data model.

![i-Scope Mobile Architecture](image1.png)

**Fig. 1: i-Scope Mobile Architecture**

The design of a client-server framework was essential to obtain good visualisation performances on mobile devices, while preserving the access to the semantic information of large CityGML models. By shifting computational tasks to the server side (e.g. content optimisation to small mobile screens or complexity reduction to satisfy certain memory requirements) we reduced the amount of data to be transferred as well as parsing times. An additional feature of the framework is the separation between the visualisation of the geometry (WMTS*) and the access to the semantic content of the CityGML (using WFS). The architecture assumes a CityGML with a geometry that does not change in real-time, since access to real-time data is minimally supported by the web standards used.

IV. Implementation

There are some conditions that have to be satisfied in order to achieve an interactive and efficient visualisation of CityGML models on mobile devices. The visualisation has to be progressive rather than loading the entire model at once, e.g. to avoid unresponsive interactions as well as memory and storage limitations. Hence, mechanisms for the creation of multiple levels of detail have to be provided, to enable large scale visualisations while balancing both the memory footprint and the amount of information that can be visualised. The support for thematic layers should be included to give users effective means to organise and access information layers. And, interactive exploration techniques should be adopted, so only the information that meets certain filters is visible.

A. Server side overview

In our framework, CityGML data models are visualised progressively on mobile devices by taking advantage of a data tile methodology, i.e., a methodology that splits the information into a pyramid of non-overlapping areas at multiple zoom levels. On the one hand, this facilitates the visualisation of large size models when using a limited network bandwidth because it reduces the amount of data to be exchanged over the network – we do not transfer the whole city model. On the other hand, it eases on mobile hardware requirements necessary to store and visualise the information.

The tile strategy implemented in our framework combines the use of the Plate Carree projection (also known as geographic projection), which maps a 2:1 rectangular image to a sphere, with an interpolation of different levels of details (LoD) of the geometry. The first level divides the world into 36 x 36 degree pieces. The number of tiles quadruples for additional levels, see Figure 2.

![Tile’s subdivision algorithm](image2.png)

**Fig. 2: Tile’s subdivision algorithm**

For performance reasons, each tile stores only an optimised version of a subset of the 3D city model (geometry only), in successively higher resolutions. We used the Wavefront OBJ format for the data transfer process because it is an open
format and it is widely used by other 3D graphics application vendors. Additionally, its geometry can be stored either in
ASCII (using the ".obj" file extension) or in binary (using the .MOD extension). Most importantly, it can be parsed with
a minimal overhead memory footprint.

LoDs of 3D buildings were mapped to layer levels heuristically based on a function that depends on the complexity of
the dataset. We did not implement algorithms to simplify (interpolate) the geometry across different layer levels and
LoDs, but since we use the OBJ format, it is straightforward to do it with existing tools. Nevertheless, we implemented a
mechanism to reduce the quality of the textures, which we identified as critical. Many 3D building have textures that are
not optimised for real-time rendering. For example, a single building can have textures that sum up to 1GB, which would
hardly render on mobile devices.

The parameters required to configure the tiling service are:
a list of tuples (LoD, Starting layer level) and a list of
tuples (Layer level, image quality). Intermediary values are
interpolated automatically by the system and 3D models are
generated from the CityGML model on as-you-need basis.
OBJ models are read-only and are regenerated when their
timestamps are older than the timestamp of CityGML model.

Pre-cached tiles are stored using the following scheme, which is directly mappable into a WMTS user request.

![Storage scheme](image)

**Dataset Name \# \# \# \# \# \# \# \# \# \# \# .abc**

- **Name of Data Set**
- **Layer Number**
- **Row**
- **Column**
- **Format**

Fig. 3: Storage scheme

The same storage scheme is used by the mobile device. Therefore, we know a-priori the data that is available for a
certain location and layer.

CityGML data modelling operations are supported via Web Feature Service (WFS) standard interface, which exposes also
methods for data discovery and query. Additional processing and transformation operations are possible via standard ser-
dices like Web Coordinate Transformation Service (WCTS)
and Web Processing Service (WPS), hence minimising the data
processing requirements on the mobile platform. Lastly, the
Web Map Tile Service (WMTS) is used to fulfil the initial
requirement: an efficient mechanism to visualise geometry.
Although the standard was designed for accessing tiled images,
nothing rules out that it can not be used to access other
data formats. As required by the standard, we have included the
Wavefront OBJ file format in the GetCapabilities request,
which serves, when requested, the geometry located within a
certain tile. CityGML feature ids are stored in the OBJ format
under the tags name objects 'o' and polygon groups 'g'. The
semantic information in the CityGML model, associated with
these ids, is then accessed via WFS on an as-you-need basis.
For the implementation of WFS and WPS, we have chosen
the GeoServer [5], which offers an easy web interface.

The sequence of requests and responses is the following.
First, the mobile client requests a tile using WMTS. The server
serves the tile, if it exists, to the client (compressed with
LZMA algorithms, together with the textures). Otherwise, the
CityGML model is loaded into the internal memory of the
server and all features corresponding to the requested tile are
extracted, optimised, and then exported to the OBJ format. A
server cache is created and used for any subsequent requests of
the same tile. From a client perspective, this approach
improves downloading times, reduces memory footprint, and
speeds up parsing operations. Optionally, the Web Coordinate
Transformation Service can be used by mobile clients that do
not support WGS84 to re-project the dataset.

### B. Mobile side overview

The steps for retrieving and interacting with CityGML models are:
first, we compute the terrain area that is visible
based on the camera field of view; second, we create a priority
list containing the tiles that are not available locally – tiles at
the centre of the screen or in the field of view have higher
priority; third, tiles from the priority list are requested to
the WMTS; finally, the tiles (a compressed OBJ compatible
enriched format with all its assets) are cached locally based
with their URL serving as a key.

In addition to the geometry, our framework provide also ac-
cess to semantic and topological information of the CityGML
model. Semantic information is accessed using the WFS
standard and the corresponding geometry id stored in the OBJ
file (tag object ‘o’ and group ‘g’), which match the feature
id in the CityGML data model. In this way, the information
associated with the geometry is accessed on an as-you-need
basis and does not have to be locally stored. The WFS protocol
could also be used to retrieve the feature’s geometry. However
such strategy was proven to be much slower.

Our mobile implementation uses a multi-scale model de-
defined by five Levels of Detail (LoD) [6] to efficiently visualise
large scale areas. LoD 0: a regional level, depicting the
footprint and roof edge representations; LoD 1: a region level,
containing block models derived by extruding ground plans;
LoD 2: a district level, with block models that include roof
geometry; LoD 3: an architectural representation that describes
the outside of the building; and a LoD 4 that extends a LoD 3
with indoor information like room divisions or furniture. LoD
3 and LoD 4 were created manually.

The mobile application renders data tiles using the following
rules. First the list of tiles to be rendered is computed using
navigation-tied parameters (e.g. the eye distance to the tile
and camera altitude). If the list of tiles does not fit the device
memory, it attempts with smaller layer levels until it fits. The
user can control the operation by enabling or disabling specific
layers of interest. Also, low resolution levels are expected to
be downloaded first, hence to be available. Often the difference
between layer levels is the quality and size of the texture to
be rendered.
To pick features from the 3D world, a key-colour picking algorithm was implemented. Although this method has linear complexity and uses the potential of GPUs, it has also some limitations that we consider irrelevant. While geometry based ray-casting permits defining and calculating collisions of any ray in world space, colour based ray-casting only works for those rays that start from the current point of view. Colour based ray-casting works by rendering the collision scene from a certain point of view. Hence, we draw each feature with a unique colour. Then we add each colour and its associated id to the picking hashtable. Once the scene is rendered, we use the id associated with the colour of the pixel in frame-buffer, matching the position of the cursor, to access the feature. For datasets containing large amount of features, this approach is known to be much faster than computational intensive octree-based approaches.

V. CONCLUSION

This paper proposes a new methodology for the modelling workflow of city data – from the generation of 3D buildings to its visualisation on the mobile devices.

The framework proposed is driven by the usage of a data model called CityGML, which was adopted not only because it is an OGC standard, but mostly because it supports the representation of urban scenes as a unified data model. Hence, it makes it interoperable and facilitates its use.

To decrease the loading time of city models, we integrated an algorithm to process a CityGML model into optimised tiles of geometry, which are then compressed and streamed to mobile devices. An intermittent geometry format is also used to decrease the memory footprint required to parse the data on the mobile device. An efficient visualisation mobile framework was archived by combining the WFS with the local intermittent geometry.

Future developments should target new techniques to make the navigation easier. More dynamic LoD’s will have to be generated to account for specific mobile devices specs.

ACKNOWLEDGEMENTS

This research has been supported by the European Commission (EC) under the projects i-Scope and Sunshine. The authors are solely responsible for the content of the paper. It does not represent the opinion of the European Community. The European Community is not responsible for any use that might be made of information contained herein.

REFERENCES