Delivering time-evolving 3D city models for web visualization
Vincent Jaillot, Sylvie Servigne, Gilles Gesquière

To cite this version:

HAL Id: hal-02501662
https://hal.archives-ouvertes.fr/hal-02501662
Submitted on 20 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Delivering time-evolving 3D city models for web visualization

Vincent JAILLOT, Sylvie SERVIGNE, Gilles GESQUIERE

Studying and planning urban evolution is essential to understanding the past and designing the cities of the future and can be facilitated by providing means for sharing, visualizing, and navigating in cities, on the web, in space and in time. Standard formats, methods, and tools exist for visualizing large-scale 3D cities on the web. In this article, we go further by integrating the temporal dimension of cities to geospatial web delivery standard formats. In doing so, we enable interactive visualization of large-scale time-evolving 3D city models on the web. A key characteristic of this article lies in the proposed four-step generic approach. First, we design a generic conceptual model of standard formats for delivering 3D cities on the web. Then, we formalize and integrate the temporal dimension of cities to this generic conceptual model. After that, we specify the conceptual model into the 3D Tiles standard at logical and technical specification levels, resulting in an extension of 3D Tiles for delivering time-evolving 3D city models on the web. Finally, we propose an open-source implementation, experiments, and an evaluation of the propositions and visualization rules. We also give access to reproducibility notes allowing researchers to replicate all the experiments.

Keywords: Data models; Spatio-temporal data modelling; 3D visualization; Open systems and standards; urban applications

1. Introduction

Cities evolve in a continuous manner on various scales in space and time (e.g. roof restoration, construction of a bridge, renovation of a district, etc.). Understanding these changes from the past and planning the future of cities is key to a number of applications such as understanding the urban fabric, representing and managing cultural heritage, performing energy simulations, etc. We, therefore, need to provide solutions to store, query, exchange, visualize, and interact with digital cities in space and time at various granularity levels.

Numerical representations of cities are usually based on models composed of sets of 3D geometric objects, thematic information (e.g., building, vegetation, address, etc.) and of hierarchical relationships between these objects (Stadler and Kolbe 2007, Kolbe 2009). These representations effectively express the geometric and thematic dimensions of the city at
various granularity levels. We refer to these representations as 3D city models throughout this paper. These 3D city models represent structural elements of the city, such as buildings, bridges, etc. They generally represent spatial areas of tens or hundreds of km² (corresponding to millions of triangles, plus associated thematic information and hierarchical relationships), introducing the issue of massive dataset management. Delivering these large-scale 3D city models on the web for their interactive visualization has been made possible due to the recent and rapidly evolving standards, such as the 3D Portrayal Service Standard (Open Geospatial Consortium 2017a), 3D Tiles (Open Geospatial Consortium, 2018a) and Indexed 3d Scene Layer (I3S) (Open Geospatial Consortium, 2017b).

Many works focus on the representation of the evolution of the city through time. Some approaches focus on visually representing the evolution of a specific spatial area (e.g., building, district, etc.) (Yano et al. 2008, Schindler and Dellaert 2012, Rizvic et al. 2015) in which the study area commonly is represented in different states in time annotated by timestamps or periods. Some represent these states by collections of 3D geometric objects, while others go further by using 3D city models. A common formalization of the temporal dimension of cities is based on adding transactions between states of city objects or between groups of city objects (Renolen 1997, Stefani 2010, Chaturvedi et al. 2017, Samuel et al. 2018). These transactions have been categorized as creation, modification, etc. and can be further characterized by more specific information; for example, why has it changed, or who made this change? Figure 1 gives an example of the evolution of a building and its modelling based on states versus states-and-transactions. The building of consideration experiences three states: the initial construction of the building, an additional floor, and an extension. A state representation of this building stores data of the three states to represent the building. Alternatively, the state-and-transaction representation includes both the three states of the building and the information regarding the transactions. Using both states and transactions
together represents the building better because the approach connects objects across the states of the building and meanwhile retains thematic information describing the changes. City models based on the state-and-transaction representation are called time-evolving 3D city models in the following.

![Evolution of a building](image)

Figure 1. Example of the evolution of a building and two possible models based on states modelling and on states and transactions modelling.

While the previous studies proposed many models for time-evolving 3D city models, methods are still lacking to efficiently deliver these large-scale, time-evolving 3D city models on the web in a manner enabling interactive spatio-temporal visualization. To address this challenge, we propose a generic approach based on standards. The proposed approach models the evolution of 3D city models' features based on states and transactions with thematic information describing the changes. In addition, the proposed approach represents changes at various levels of granularity in space and time. Furthermore, the proposed approach enables visualization of large-scale data sets (e.g. at a city scale) and allows users to navigate through
the 3D city interactively in space and time. The development of the proposed approach leads to the following contributions and outline:

- **Developing a generic conceptual model** for delivering 3D cities for web visualization based on standard (section 3.1)
- **Formalizing** and **integrating the temporal dimension** of cities to this model (section 3.2)
- **Specifying** this model at the logical and technical levels by formalizing and extending the 3D Tiles standard (section 4)
- **Implementing** the proposed approach in **open-source web software**, **evaluating** the software’s performance and **documenting** the procedures to ensure **reproducibility** (section 5)

In the following section, we examine state of the art on delivering 3D city models on the web (section 2.1) and on representing the temporal dimension of urban data (section 2.2). Then, we present a summary which highlights the limits of current state of the art. (section 2.3).

2. **State of the art**

2.1. **Delivering 3D city models on the web**

Prieto *et al.* (2012) adopts the X3D format for delivering 3D city models on the web. However, the X3D format considers only the geometry of city models, discarding the thematic information and hierarchical relationships. Other initiatives have proposed the delivery of CityGML (Open Geospatial Consortium 2012) data on the web (Gesquière and Manin 2012, Pispidikis and Dimopoulou 2018) but fall short of optimal delivery when used by web clients due to two key reasons. First, CityGML is designed for data modelling and
exchange (Kolbe 2009), not visualization or user interaction which are important for web-client applications. In addition, CityGML is not based on web-friendly data formats (e.g. JSON), which makes it harder to use in a web context.

The Open Geospatial Consortium (OGC) proposes a set of web services to deliver geospatial data on the web (WFS\(^1\), WMS\(^2\), etc.). While, WFS could be used to deliver 3D city models on the web, WFS is unsuitable for large-scale 3D data sets (Open Geospatial Consortium, 2018b). Later, two community standards have been proposed for delivering heterogeneous 3D geospatial data sets: 3D Tiles (Open Geospatial Consortium, 2018a) and I3S (Open Geospatial Consortium, 2017b). These standards share concepts and characteristics: represent the geometry (3D polygonal models or 3D point clouds) and thematic information of geospatial data (Open Geospatial Consortium 2018b); use spatial indexing methods to organize large data sets; and adopt JSON and binary representations for efficient web-based operations. The main differences between these standards lie in their ways of representing data at the technical specification level. For instance, I3S has a unified way of representing features of the city whereas 3D Tiles include different formats: Batched 3D Model (b3dm) and Instanced 3D Model (i3dm) (which are based on glTF) for 3D polygonal models and Point Cloud (pnts) for 3D point clouds. Schilling et al. (2016) extends the 3D Tiles with the hierarchical relationships of 3D city models to enable the delivery of complete 3D city models (geometry, thematic and hierarchical relationships) However, neither 3D Tiles nor I3S integrates the temporal dimension of cities.

In addition to these formats, the 3D Portrayal Service OGC standard provides an interface for 3D geospatial data to support queries about a 3D scene in a web context with a generic and standard approach. Responses to a query may be in any format including 3D

\(^1\) https://www.opengeospatial.org/standards/wfs
\(^2\) https://www.opengeospatial.org/standards/wms
Tiles and I3S (Koukofikis et al. 2018). Hence, either 3D Tiles or I3S can provide a common interface for delivering 3D city models on the web.

2.2. Representing the temporal dimension of urban data

Chaturvedi and Kolbe (2019) identify two categories of temporal changes related to 3D city models. The first type of changes concerns the evolution of structural elements of the city (e.g., building, bridges, etc.). The second type of change considers properties external to city models and the property’s varying values over time (e.g., time series such as solar irradiance or electricity consumption of a building).

With the growing need for modelling and visualization of a city and the city’s evolution over time, researchers have proposed a 4D (3D geometry and time) digital reconstruction of a specific monument (Rizvic et al. 2015), district (Schindler and Dellaert 2012) or city (Yano et al. 2008) along with visualization methods and tools. However, these proposals represent the temporal dimension as a collection of states of the dataset (i.e., independent representations of the data at different timestamps or for different periods of time, c.f. representation of states in Figure 1). As such, none of the proposals considers the thematic information or the hierarchical relations between objects, which are a key element for numerous applications (Döllner et al. 2006, Biljecki et al. 2015).

Other formalizations are based on modelling the states of geospatial objects and the transitions across these states. Renolen (1997) proposes a formalization of six possible transitions occurring to geospatial features (e.g., buildings). Figure 2 illustrates these six basic types of transitions. In this formalization, a transition can either consist in a creation (e.g., construction of a new building), alteration (e.g., roof restoration), cessation (e.g., bridge destruction), reincarnation (e.g., destruction and reconstruction of building), merge (e.g., two buildings are joined together to form one building) or split (e.g., a building is separated into
two parts to form two distinct buildings). Stefani (2010) extended these types of transitions from an architectural and urban point of view, giving seventeen possible types of transitions between two states of a heritage building.

![Diagram of six basic types of transitions between geospatial objects](image)

Figure 2. Six basic types of transitions between geospatial objects (Renolen 1997).

Chaturvedi *et al.* (2017) extends the CityGML standard to represent the temporal dimension of 3D city models to represent the states of geospatial features and the transitions between these states (called *Transactions*). As this work is currently being standardized for CityGML 3.0, we will also use the term *transaction* to conform with the terminology used in Renolen (1997) and Stefani (2010). There are other related terms suggested in the literature. Chaturvedi *et al.* (2017) introduce the concept of *Versions* to and *VersionTransitions*. A *Version* is a collection of temporal geospatial features (i.e. having a time-span). A *VersionTransition* is a collection of *Transactions* where each *Transaction* expresses the changes that relate two geospatial features. In addition, *Transactions* can refer to changes at various levels of spatial granularity, which is one of the requirements in our development of time-evolving 3D-City web delivery and visualization. For instance, a transaction can refer to a building modification, a roof restoration, or a door change, etc. However, the method by Chaturvedi *et al.* (2017) is based on CityGML, which is unfit for data delivery and interactive
visualization on the web, as explained in section 2.1. In addition, Chaturvedi et al. (2017) consider only three types of Transactions are possible: insert, delete, and replace, whereas Renolen (1997) and Stefani (2010) have proved that more transactions are needed to fully describe the evolution of city components.

Samuel et al. (2018) extend the work by Chaturvedi et al. (2017) to represent various possible scenarios of the evolution of a city and to store, query, and update versions, version transitions and transactions of scenarios of city evolution. However, this solution emphasizes managing the evolution of the city (e.g., to enable historians to update information on the city’s evolution), not delivering data on the web for visualization and interaction.

2.3. Summary

The goal of the work presented in this paper is to deliver large-scale, time-evolving 3D city models on the web in a way allowing for interactive spatio-temporal visualization, with an approach based on standards. Time-evolving 3D city models have geometric (3D), thematic and temporal (states and transactions) dimensions with hierarchical relationships among objects. None of the standards and methods reviewed in this section integrate all these aspects. However, the three established standards (CityGML, 3D Tiles and I3S) present some of these characteristics. They are compared with our needs in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Geometric dimension</th>
<th>Thematic dimension</th>
<th>Hierarchical Relationships</th>
<th>Temporal dimension</th>
<th>Interactive visualization on the web</th>
</tr>
</thead>
<tbody>
<tr>
<td>CityGML</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>3D Tiles</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>I3S</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 1. Comparison of CityGML, 3D Tiles, and I3S according to our needs for delivering time-evolving 3D city models on the web.
While CityGML can model all the dimensions of time-evolving 3D city models, the methods to deliver CityGML data on the web are unsatisfactory (see section 2.1). We, therefore, focus our work on 3D Tiles and I3S. Although they do not integrate all the aspects of time-evolving 3D city models, extensions are possible to meet our needs. Given the conceptual similarities between 3D Tiles and I3S, we first propose a generic conceptual model in UML (Gen3DCity model, section 3.1) based on the two standards so that our proposed approach can be implemented on either of the format, I3S or 3D Tiles. We then propose a formalization of the temporal dimension of cities and integrate the formal temporal model into Gen3DCity (Gen4DCity model, section 3.2). This formalization is based on the concepts proposed by Renolen (1997), Stefani (2010) and Chaturvedi et al. (2017). However, we modify and extend these concepts to propose a more generic formalization of the temporal dimension. After that, we specify Gen4DCity at logical and technical specification levels into the 3D Tiles standard (section 4). Table 1 shows that 3D Tiles format is more suited to our requirements than I3S since an extension of 3D Tiles\(^3\) can represent hierarchical relations of 3D city models (Schilling et al. 2016). However, the first two steps of generic conceptual modelling are able to map our model to I3S.

\(^3\) See extension/3DTILES_batch_table_hierarchy/ folder of the following repository:

https://archive.softwareheritage.org/swh:1:rev:50ca4c646da460395b0a06ed85d7f54a727e09a5;origin=https://github.com/AnalyticalGraphicsInc/3d-tiles/
3. Generic UML models

3.1. Conceptual modeling for delivering 3D city models on the web generalizing 3D Tiles and I3S

Figure 3 presents **Gen3DCity**: a generic conceptual model for delivering 3D city models on the web, that is to say that we use **UML Class Diagrams at Conceptual level and not at Specification or Implementation level** (Embley 2011, Fowler 2003). This model is an abstraction of the 3D Tiles and I3S standards. The names of the entities are independent of 3D Tiles and I3S and have been chosen to match their concepts with an effort to be as generic as possible.

![UML Diagram]

Figure 3. **Gen3DCity**: a generic UML conceptual model for delivering 3D city models on the web, generalizing 3D Tiles and I3S standards.

*Geospatial Features* (e.g., building, roof, bridge, etc.) can have a *Geometry* (e.g., point cloud, polygonal model, etc.) and *Thematic Attributes* (e.g., owner). These *Geospatial Features* are organized according to a spatial indexing method (e.g., quadtree, octree, etc.).

The resulting index is represented by a set of related *Spatial Nodes* (that is to say nodes computed according to spatial criteria), most frequently representing a tree. 3D Tiles and I3S both allow multiple indexes that can be grouped to a root object. This root object is represented by a *Set* in our model. In the case of 3D Tiles, a *Spatial Node* can either directly
reference a set of Geospatial Features or another new index (i.e., another Spatial Node hierarchy). In the case of I3S, indexes are grouped into layers (representing a data layer, e.g., a building layer), which constitute a scene. Finally, 3D Tiles and I3S can represent the geospatial data at various levels of details in a Spatial Nodes hierarchy: a child Spatial Node can be a refinement of its parent.

The representation of geospatial features, spatial nodes, sets, and spatial-nodes hierarchy expresses the shared concepts of 3D Tiles and I3S and can work with the two standards at the conceptual level (e.g., to extend them) without being tied to one or the other. Moreover, the representation can support implementation choices in either standard. In the following section, we formalize the temporal dimension of cities and integrate it into this representation.

3.2. Integration of the temporal dimension for delivering time-evolving 3D city models on the web

Figure 4 presents Gen4DCity that extends Gen3DCity (in white) with our proposed formalization of the temporal dimension (in gray). The appendix provides additional information about data types and enumerations used in this model and other models discussed in this paper.
Figure 4. Gen4DCity: A generic UML conceptual model for delivering time-evolving 3D city models on the web based on 3D Tiles and I3S standards. Entities from Gen3DCity (from Figure 3) appear in white, and entities for temporal information manifest in grey.

Extensions to Gen3DCity take four steps. First, we add a *Time-span* entity specifying the interval of existence between two dates in the real world for each *Geospatial Feature* to note the feature’s valid time (e.g. a building built in 1975 and destroyed in 2012). *Spatial*
Nodes can also be associated with a *Time-span* to use spatio-temporal indexing methods (Theodoridis *et al.* 1998, Hadjieleftheriou *et al.* 2002, Mokbel *et al.* 2003) or temporal indexing methods (Bertino *et al.* 2012). Organizing spatio-temporal features using a spatial indexing method is easy to implement but may not be optimal. We decide to leave the choice of type of indexing method (spatial, spatio-temporal or temporal) to the user, as the user preference is highly dependent on the data and on the application use case. A *Set* can also be associated with a *Time-span* for the entire collection of spatial nodes.

Secondly, we introduce the *Transaction* entity to represent the changes between geospatial features. A *Transaction* is composed of a *description* and *tags* and is subject to two specifications: *PrimaryTransaction* and *TransactionAggregate*. The idea behind this proposition is to combine primary transactions for complex transactions. *PrimaryTransaction* entities have a *type* attribute which can have one of the following five values: *creation*, *demolition*, *modification*, *union* or *division*. We consider these five types as primary types, individually or in combination, capable of modeling other types of transactions encountered in the literature. For instance, Renolen (1997) proposes six types of transaction between geospatial features (Figure 2), five of which corresponds to the primary types we propose. However, the sixth one (*reincarnation*) can be modeled using a *TransactionAggregate* composed of two *PrimaryTransactions* of types *demolition* and *creation* (Figure 5).

*PrimaryTransactions* and *TransactionAggregates* adequately capture all the transactions types proposed by Stefani (2010). The proposed treatment of transactions has two main advantages. First, the proposed treatment is more generic than the current solutions because it doesn’t restrict the transactions to a limited set of predefined types. Secondly, the proposed treatment of transactions can represent *Transactions* at different levels of detail, such as the *reincarnation* type proposed by Renolen (1997) at a macro level of detail.
(TransactionAggregate) or at a finer grain, with PrimaryTransactions of the types demolition and creation (Figure 5).

![Diagram of Transaction Aggregate and Primary Transactions](image)

**Figure 5.** An example of aggregated transactions.

In the fourth step of the extensions to Gen3DCity, we introduce the Version entity to group geospatial features for two purposes. First, a Version entity groups geospatial features with hierarchical relationships (e.g. a building, its roof and its walls) to form a version of the collection. The second purpose is to express concurrent versions of a group of geospatial features through an evolution. For instance, we may have multiple proposals to renovate a district. A version can be created to represent individual possible outcomes of these projects. Then, the VersionTransition entity can be used to group all the Transactions that happened between these Versions. A VersionTransition also has a type defined in the TransitionValue enumeration.

This formalization of the temporal dimension has two main advantages compared to existing solutions. First, the formalization can represent the evolution of the city over time on different spatial and thematic scales (city level, geospatial features level) and covers many types of Transactions among Geospatial Features. The second main advantage is flexibility; we can represent the evolution of a city at various degrees of precision depending on data and needs. For example, we can only use the Time-span entity to represent different states of Geospatial Features and of the Set and use a spatio-temporal or temporal indexing method to organize large datasets. Alternatively, we can use the Transaction entity to represent the
transactions between Geospatial Features. Another option is to use all the entities formalizing the temporal dimension in this model to group Geospatial Features together in Versions and represent the transitions between these Versions with VersionTransitions.

In section 4, we specify Gen4DCity at the logical and specification levels in the 3D Tiles standard format.

4. Specification into 3D Tiles

4.1. 3D Tiles logical model and temporal extension

Figure 6 presents both Logic3DCity and Logic4DCity. Logic3DCity (in white) is a proposition of logical model of the 3D Tiles standard. It depicts the current state of 3D Tiles (OGC Community Standard version 1.0) and covers all the data modelling part of the specification. Logic3DCity is a specification of Gen3DCity. Logic4DCity is a logical model for delivering time-evolving 3D city models on the web. It integrates the formalization of the temporal dimension to Logic3DCity. Logic4DCity is a specification of Gen4DCity. The recursive composition association of Feature (thick line) is a proposition of conceptual representation of hierarchical relationships between features of 3D city models (i.e. a conceptual representation of the batch_table_hierarchy extension of 3D Tiles).

---

5 See extension/3DTILES_batch_table_hierarchy/ folder of the following repository:
https://archive.softwareheritage.org/swh:1:rev:50ca4c646da460395b0a06ed85d7f54a727e09a5;origin=https://github.com/AnalyticalGraphicsInc/3d-tiles/
Figure 6. **Logic4DCity**: UML logical model for time-evolving 3D city models on the web based on extending 3D Tiles. A logical model of 3D Tiles, **Logic3DCity**, is proposed in white and the temporal extension is represented in gray. The recursive composition association of Feature (thick line) is a conceptual representation of the *batch_table_hierarchy* extension of 3D Tiles.

3D Tiles is composed of a Tileset (corresponding to Set of Gen3DCity). Asset's attributes *version* (version of 3D Tiles) and *tilesetVersion* (identifier for tracking updates of a
Tilesets are not related to representing the evolution of the city over time. A Tileset is composed of a set of Tiles (corresponding to Spatial Nodes of Gen3DCity). Each Tile has up to two BoundingVolumes: boundingVolume and viewerRequestVolume. The viewerRequestVolume makes it possible to specify a volume where the tile must be displayed (i.e. the content of the tile is displayed only if the camera is inside this volume). Each Tile has a TileContent that can also have a BoundingVolume, which must closely fit the features of the tile. The TileContent references either a Tileset or a FeatureSet, which is a collection of Features plus some FeatureSetProperty (e.g. number of points, offset, scale, etc.). The FeatureSet entity depicts a generalization of the formats proposed by the 3D Tiles community to represent features: Batched 3D Model (b3dm), Instanced 3D Model (i3dm), Point Cloud (pnts) and Composite (cmpt). A Feature (corresponding to Geospatial Feature of Gen3DCity) can have a FeatureGeometry (Geometry of Gen3DCity). It can also have FeatureProperties (Semantic Attribute of Gen3DCity) such as application-specific attributes (e.g. the owner of a building) or appearance information (e.g. color or texture coordinates). A FeatureGeometry can belong to several Features, so it is possible to have a 3D model instanced several times (e.g. with different positions or different scales such as in the i3dm format). At the technical specification level, a FeatureProperty is stored in a batch table or in a feature table depending on the property and on the format (b3dm, i3dm, etc.).

We integrate the formalization of the temporal dimension proposed in Gen4DCity with the entities in gray. Most of the concepts have been integrated by direct mapping from Gen4DCity thanks to the abstract modelling work done in section 3.1 and to the mapping of Gen3DCity to Logic3DCity described above. VersionTransition, Version and Transaction are aggregated in the Tileset since they are related to Features across the Tileset (i.e. that can be in different Tiles). The Spatial Node’s Time-span (from Gen4DCity) is aggregated in the BoundingVolume and not in the Tile (which could have been done by direct mapping since
Spatial Node is mapped to Tile. This way not only can the boundingVolume attribute of the Tile have a Time-span, but also its viewerRequestVolume and the boundingVolume attribute of TileContent. The boundingVolumes make it possible to index Features using spatio-temporal indexing methods. The viewerRequestVolume makes it possible to show the content of a Tile only when the display date (date at which the data is displayed to the user) is inside its Time-span.

In section 4.2, we propose an extension of 3D Tiles for time-evolving 3D city models at technical specification level. We name it 3DTiles_temporal (following the naming convention of 3D Tiles).

4.2. Extension at technical specification level

3D Tiles contains concepts named extension and extra. Figure 7 is a proposition of description of these concepts. Extension makes it possible to extend 3D Tiles entities. It has a name and it is composed of an application-specific Object. An Object can be composed of other Objects and/or Attributes. Extensions can be aggregated in an ExtensionSet. An ExtraSet is a collection of Extras. An Extra is an application-specific Attribute, making it possible to add specific metadata to 3D Tiles entities. All the entities of the 3D Tiles specification can have an ExtensionSet and an ExtraSet.
Figure 7. Description of the concepts for extending 3D Tiles as a UML conceptual model.

The 3DTiles_temporal extension specification is described with JSON schemas, which we make available online (see Data and codes availability statement). It uses the Extension concept. This way, the core of the standard is not modified, but there is still an impact on users of 3D Tiles.

The specification is composed of three Extension objects respectively aggregated in Tileset, BoundingVolume and BatchTable. The Tileset has a startDate and an endDate and stores versions, versionTransitions and transactions. The extension of the BoundingVolume also has a startDate and an endDate to allow the representation of 4D bounding volumes. The BatchTable is where the thematic attributes of features are stored. We extend it with two arrays representing the time of existence of each Feature, namely startDates and endDates. The BatchTable also holds an array of identifiers of Features enabling them to be identified in transactions and versions. This specification results from models previously presented and more specifically from Logic4DCity.

The models proposed in this section allow to model the temporal dimension of 3D city models by extending the 3D Tiles standard format. In particular, they allow to keep track of the evolution of 3D city models' features at various granularity levels and to add thematic
information to these, which was some of the challenges stated in the introduction of this paper.

5. Implementation and evaluation

In this section we present an implementation allowing to deliver and visualize time-evolving 3D city models on the web with the 3DTiles_temporal extension proposed in this paper. We start with a presentation of the software implementation (section 5.1). We continue with an evaluation of our propositions (section 5.2), demonstrating that they allow to achieve interactive spatio-temporal visualization of large-scale time-evolving 3D city data sets on the web. We finish with the proposition of visualization rules for urban evolution visualization and apply them for the visualization of a time-evolving 3D city model (section 5.3). Complete notes to reproduce the results of this section are provided in the Data and codes availability statement.

5.1. Software implementation

We implemented the temporal extension of 3D Tiles proposed in this paper in a 4-tier web architecture. This implementation is fully open source and allows for the creation, delivery and visualization of time-evolving 3D city models on the web. It is composed of two main processes.

The first process enables the creation of time-evolving 3D city models in 3D Tiles with the temporal extension proposed in this article. It is implemented in Py3DTiles⁶ which is a software component making it possible to convert geospatial data to 3D Tiles. This process is composed of two parts. The first one is an implementation of the temporal extension

⁶https://archive.softwareheritage.org/swh:1:dir:2607d3af311abef046958fca57ad2f2f4d929a02;origin=https://github.com/VCityTeam/py3dtiles/
proposed in this article and it follows the technical specification of the extension. The second part (*City Temporal Tiler*⁷) allows to go from 3DCityDB (Yao et al., 2018) databases holding timestamped city models and from a list of changes between the buildings of these city models to a 3D Tiles + temporal extension dataset. The list of changes between the buildings of timestamped city models can be computed with an algorithm allowing to detect geometric changes between buildings at different timestamps (Pedrini et al., 2015). This process is implemented in the 3DUSE⁸ software component. It is not detailed here as it is more of a preprocessing step to obtain a dataset rather than directly allowing to prove the validity of the contributions proposed in this article.

The second process enables delivery and visualization of 3D Tiles + temporal extension datasets on the web. Delivering these time-evolving 3D city models can be done using any web server such as an http web server for instance. The client-side processing of the dataset has been implemented in UD-Viz⁹ (Urban Data Visualization). UD-Viz is a web client for geospatial data visualization and navigation, based on the geospatial data visualization framework iTowns¹⁰.

Since the formalization of the temporal dimension has been integrated to 3D Tiles as an extension (following the recommendations of the standard), it could be implemented into other software components supporting 3D Tiles (e.g. Cesium¹¹). This software implementation allows to run experiments for evaluating our contributions (section 5.2).

---

⁷https://archive.softwareheritage.org/swh:1:cnt:189a8695b5e3ea8438499206f88a83e74543ddcd;origin=https://github.com/VCityTeam/py3dtiles/
⁸https://archive.softwareheritage.org/swh:1:rev:14b1e3960962dbb9732d13ec64e583192b452242;origin=https://github.com/VCityTeam/3DUSE/
¹⁰http://www.itowns-project.org/
¹¹https://cesiumjs.org/
5.2. Experiments and evaluation

The objectives of this section are to evaluate the scalability and the interactivity of the 3DTiles_temporal extension. In particular, we show that it allows to visualize large-scale time-evolving 3D city models while allowing interactive spatio-temporal navigation. First, we present the data set and the different standard representations chosen for this evaluation (section 5.2.1). Secondly, we evaluate the interactivity of the spatio-temporal navigation (section 5.2.2).

5.2.1. Datasets comparison

In order to evaluate the 3DTiles_temporal extension, we use a data set representing the buildings of the city of Lyon, France (48 km²) between 2009 and 2015. We stored this data set in three different ways: in CityGML, in 3D Tiles and in 3D Tiles extended with 3DTiles_temporal. We name these representations DS-CityGML, DS-3DTiles and DS-3DTiles-Tmp respectively. The evaluation takes place on DS-3DTiles and DS-3DTiles-Tmp. DS-CityGML is the input data used to create them and it is only added in the comparison of this section for information purposes.

DS-CityGML is composed of three independent snapshots of the city of Lyon (2009, 2012 and 2015). The original data has been downloaded from the Grand Lyon open data website\footnote{https://data.grandlyon.com/}. However, it has some quality issues. In particular, it has syntax errors, the data structure and the version of CityGML are not the same between vintages (2009, 2012, 2015). A pre-processing step allowed to correct these issues. In addition, DS-CityGML has been simplified by removing textures coordinates and generic thematic attributes that are not included in DS-3DTiles and DS-3DTiles-Tmp.
**DS-3DTiles** is composed of the same three independent snapshots (2009, 2012 and 2015) of the buildings of the city of Lyon than **DS-CityGML**. It has been created by converting each snapshot of **DS-CityGML** into the 3D Tiles format using the **City Tiler** process\(^\text{13}\) of **py3DTiles** (transforming CityGML data sets to 3D Tiles).

**DS-3DTiles-Tmp** represents the buildings of Lyon between 2009 and 2015 but stored as a time-evolving 3D city model in 3D Tiles extended with **3DTiles_temporal**. It contains states of buildings along with time-spans and transactions between these states. It has been computed with the **City Temporal Tiler** of **py3DTiles** (presented in section 5.1). The input data are **DS-CityGML** and a list of changes between its vintages.

Table 2 compares the size in Megabytes of **DS-CityGML**, **DS-3DTiles** and **DS-3DTiles-Tmp**, as well as the number of buildings they store.

<table>
<thead>
<tr>
<th></th>
<th>Size (MB)</th>
<th>Number of buildings</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS-CityGML</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>1100</td>
<td>14827</td>
</tr>
<tr>
<td>2012</td>
<td>1100</td>
<td>14835</td>
</tr>
<tr>
<td>2015</td>
<td>976</td>
<td>24289</td>
</tr>
<tr>
<td>Total</td>
<td>3176</td>
<td>53951</td>
</tr>
<tr>
<td>DS-3DTiles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>182</td>
<td>14827</td>
</tr>
<tr>
<td>2012</td>
<td>183</td>
<td>14835</td>
</tr>
<tr>
<td>2015</td>
<td>261</td>
<td>24289</td>
</tr>
<tr>
<td>Total</td>
<td>626</td>
<td>53951</td>
</tr>
<tr>
<td>DS-3DTiles-Tmp</td>
<td>435</td>
<td>36975</td>
</tr>
</tbody>
</table>

Table 2. Comparison of a data set representing the buildings of the city of Lyon, France (48 km\(^2\)) between 2009 and 2015, stored in three different ways: in CityGML (DS-CityGML), in 3D Tiles (DS-3DTiles) and in 3D Tiles extended with **3DTiles_temporal** (DS-3DTiles-Tmp).

Table 2 shows that the size of **DS-3DTiles-Tmp** is 7.3 times smaller than **DS-CityGML** and 1.4 times smaller than **DS-3DTiles**. The difference between the CityGML-based

\(^{13}\text{https://archive.softwareheritage.org/swh:1:cnt:489e2b47720e77dc80b5edad89ab175dba30f6ef;origin=https://github.com/VCityTeam/py3dtiles/}
representation (DS-CityGML) and the 3D Tiles based representation (DS-3DTiles and DS-3DTiles-Tmp) has two main explanations. First, DS-3DTiles and DS-3DTiles-Tmp do not store the hierarchical structure of the data present in DS-CityGML. It could be added using the batch_table_hierarchy extension of 3D Tiles. Secondly, the representation of the data proposed by 3D Tiles and based on JSON and binary encoding is lighter than the xml-based representation of CityGML. One can also note that while the size of the 2012 vintage of DS-CityGML is higher than the 2015 vintage it is the opposite for DS-3DTiles. This is due to the fact that the 2009 and 2012 vintages of DS-CityGML are stored in CityGML version 1.0 while the 2015 vintage is stored in CityGML version 2.0. However, the version of 3D Tiles is the same for all the vintages of DS-3DTiles. In addition, the 2015 vintage represents buildings at a higher level of detail than the 2009 and 2012 vintages. Therefore, the size of the 2015 vintage is bigger than the size of the 2012 vintage in DS-3DTiles.

Comparing the size of DS-3DTiles and DS-3DTiles-Tmp shows one advantage of using the temporal extension proposed in this paper. The model proposed for our extension indeed enables a significant gain in size while allowing the storage of more information describing the evolution of the city (time-spans and transactions between features in this case) in comparison with CityGML and 3D Tiles. The size gain is correlated with the reduction of the number of stored buildings. It indeed falls from 53951 to 36975 between DS-3DTiles and DS-3DTiles-Tmp (which represents a ratio of 1.5). It is explained by the fact that a building having a persistent state in more than one timestamp is stored multiple times in DS-3DTiles but only one time in DS-3DTiles-Tmp. For instance, a building which is in the same state in 2009, 2012 and 2015 is stored three times in DS-CityGML and DS-3DTiles (once in 2009, once in 2012 and once in 2015) and only one time in DS-3DTiles-Tmp (with the time-span [2009;2015]).
This memory gain is a great improvement, as reducing the size of the dataset makes it possible both to reduce the memory footprint for the client and the network transmission time, which are limiting factors for scalability.

5.2.2. Spatio-temporal navigation evaluation

We recorded the time for loading, rendering and navigating in DS-3DTiles and DS-3DTiles-Tmp to evaluate the interactivity of the spatio-temporal navigation. To this end, they have been visualized in UD-Viz. The following experiments have been conducted with an Intel Core i7-4790K CPU @ 4.00GHZ and with 32GB of RAM. The data have been served using a local web server and visualized with a Firefox 67.0 web browser.

Table 3 presents the mean time as well as the standard deviation for loading and rendering DS-3DTiles and DS-3DTiles-Tmp, measured on 10 samples. It shows that the time for loading and rendering DS-3DTiles-Tmp is lower than the time for loading and rendering all snapshots (2009, 2012 and 2015) of DS-3DTiles. Moreover, DS-3DTiles-Tmp stores additional thematic information about the evolution of features in time (transactions and features’ time-span) in comparison with DS-3DTiles.

<table>
<thead>
<tr>
<th>DS-3DTiles</th>
<th>Mean loading and rendering time (seconds)</th>
<th>Standard deviation (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2009</td>
<td>1.4</td>
<td>0.03</td>
</tr>
<tr>
<td>2012</td>
<td>1.4</td>
<td>0.04</td>
</tr>
<tr>
<td>2015</td>
<td>1.9</td>
<td>0.1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>4.7</strong></td>
<td><strong>N.A.</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DS-3DTiles-Tmp</th>
<th>Mean loading and rendering time (seconds)</th>
<th>Standard deviation (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td>0.04</td>
</tr>
</tbody>
</table>

Table 3. Time for loading and rendering DS-3DTiles and DS-3DTiles-Tmp

Figure 8 presents a comparison of the animation time when navigating between timestamps with DS-3DTiles and DS-3DTiles-Tmp. Each box represents a state at a given time indicated inside the box (Ø indicating that nothing is displayed). The arrows express a change of state. The mean loading and rendering time (seconds) are respectively in light gray.
for DS-3DTiles and in black for DS-3DTiles-Tmp (respectively indicated as xx seconds / xx seconds). The temporal extension introduces a small overhead for displaying the first state (going from nothing to 2009 or 2012 or 2015). However, navigation in time is significantly improved since it only takes 0.1 seconds to switch from one state to another instead of 1.4 and 1.9 seconds with DS-3DTiles.

Figure 8. Animation time comparison of DS-3DTiles and DS-3DTiles-Tmp.

In the current implementation, the data is indexed according to a kd-tree. We believe that implementing a spatio-temporal indexing method, such as an HR-Tree (Nascimento and Silva, 1998) for instance, may reduce the time for loading and rendering DS-3DTiles-Tmp, especially for displaying the first state (from nothing displayed to a first state). Such an indexing method would indeed allow to better organize the features along the temporal axis and therefore reduce the number of features that are transmitted to the client for the first display state. In addition, it would also allow to implement a promising approach to deliver the temporal evolution as a progressive stream.

Finally, our temporal extension allows to access information between the three temporal states of DS-3DTiles (2009, 2012 and 2015). This idea is depicted in Figure 9 where we can see that the temporal extension allows to navigate in transitory states between two temporal snapshots. The measured time on this figure show that this navigation can be done at
a fraction of the cost than with classic 3D Tiles. Therefore, our extension allows to greatly enhance navigation in time.

Figure 9. Temporal navigation comparison between 3D Tiles and 3D Tiles with temporal extension.

5.3. **Visualize urban evolution in 3D and on the web**

In this section, we propose visualization rules for displaying time-evolving 3D city models represented in 3D Tiles extended with `3DTiles_temporal`. The purpose of these visualization rules are to highlight the changes in the city to help understanding urban evolution. Our proposition is to represent features' known states in a reference color (light gray in this example) and to showcase transactions between these states. Generally, transactions span for a certain amount of time and we don't have access to the geometry of the feature(s) concerned by the transaction during this amount of time. Therefore, we propose the following display rule for transactions: during the first half of the transaction, the geometry of the previous known state is displayed. During the second half of a transaction, the geometry of the following known state is displayed. The type of transaction (`creation`, `modification`, etc.) determines the color in which the geometry is displayed. In this example, only `creations`,
modifications and demolitions are managed. However, other transactions could easily be added by following the same rules. Creations are in green, modifications are in yellow and demolitions are in red. Finally, the opacity varies during the time of the transaction in the way that it is equal to 1 at the beginning and at the end of the transaction and to 0 at the half duration of the transaction. Between these timestamps, it decreases on the first half of the transaction (from 1 to 0) and increases during the second half of the transaction (from 0 to 1). Figure 10 illustrates this proposition on three types of transactions.

![Image of visual representation of buildings known states (state i and state i+1) and of transactions between these states. States are in gray, creations in green, modifications in yellow and demolitions in red.](image)

These visualization rules have been implemented in the UD-Viz web client. This implementation also allows users to navigate freely in space (3D scene) and in time (with a time slider). Figure 11 shows two screenshots of a part of DS-3DTiles-Tmp (one district of the city) in 2013 (on the left side) and in 2014 (on the right side).
Figure 11. Screenshots showing the evolution of a district of the city of Lyon (in 2013 on the left side and in 2014 on the right side) visualized in UD-Viz and using 3D Tiles extended with 3DTiles_temporal.

In Figure 11, we can clearly identify changes occurring at these two dates. In particular, we can see buildings that are being destroyed (in red), modified (in yellow) and constructed (in green). Figure 12 shows a visualization of the full city of Lyon in 2014. Visualizing the evolution of the city at this scale can be very useful to detect areas where a lot of changes happen for instance. One can then zoom to these areas to see more details about the changes (such as in Figure 11).
Figure 12. Screenshot of the visualization of the city of Lyon in 2014 in UD-Viz, using 3D Tiles extended with 3DTiles_temporal.

In this section, we showed a possible use case of the extension we proposed in this paper. Other visualization of the data may be implemented to suit specific user needs since it is independent from the data modelling part. However, this visualization can be useful to understand and share urban evolution which is of big interest for applications such as cultural heritage or urban planning.

6. Conclusion

Modelling and visualizing the evolution of a city is important for understanding and analyzing urban changes. In this paper, we propose a method for modelling, delivering and visualizing the evolution of cities on the web. First, we proposed a generic conceptual model of standard formats for delivering 3D city models on the web (Gen3DCity). Then, we proposed Gen4DCity, a conceptual model for delivering time-evolving 3D city models, extending Gen3DCity with a formalization of cities' temporal dimension. Then, we proposed Logic4DCity, a logical model extending 3D Tiles for time-evolving 3D city models. This step brought us to propose a logical model for 3D Tiles (Logic3DCity). We also proposed a technical specification of the 3D Tiles extension (3DTiles_temporal). These models allow to keep track of the evolution of 3D city models' features at various granularity levels and to add thematic information to these changes. After that, we provided an open source implementation of this extension. This implementation allowed to evaluate the interactivity of the spatio-temporal navigation in large-scale time-evolving 3D city models. In particular, we demonstrated that our contribution allows to significantly improve temporal navigation. Finally, we proposed rules of visualization for highlighting urban evolution and presented an example of visualization of the city of Lyon, France. These contributions have been made
followed by the generic approach proposed in section 1 and reproducibility notes are provided to replicate the results.

In the future, mapping of the formalization of the temporal dimension to the I3S standard may be studied in depth. Another lead would be to extend the work on data indexing. Geospatial web delivery standards allow to organize the data according to spatial indexing methods (Azri et al. 2013). The temporal extension proposed in this paper makes it possible to use spatio-temporal indexing methods (Theodoridis et al. 1998, Hadjieleftheriou et al. 2002, Mokbel et al. 2003) or temporal indexing methods (Bertino et al. 2012). It would then be interesting to study the impact of these indexing methods on querying and visualizing 3D tiles with 3DTiles_temporal extension datasets. We for instance expect historical R-Trees (Nascimento and Silva 1998) to be an efficient indexing method in this case. Finally, this work focuses on the temporal dimension of 3D city models. An interesting lead for the future may be to study an extension of this work to other types of geospatial data supported by geospatial web delivery standard formats.

**Data and codes availability statement**

A data model at technical specification level (JSON schemas) is available in figshare.com at https://figshare.com/s/9af2e1f30116ad1247e. A dataset representing the evolution of the first borough of Lyon from 2009 to 2015 using the temporal extension of 3D Tiles presented in this article is available in figshare.com at https://figshare.com/s/f06705add7ea502d276e. Detailed notes for reproducing the results of section 5 are available in the following repository:

https://archive.softwareheritage.org/swh:1:rev:96c102a30b2278c5f863f0bc1a5d66122d7d622d;origin=https://github.com/VCityTeam/UD-Reproducibility/. In particular, for section 5.2.1, refer to Articles/2020-IJGIS-Temporal/DatasetComparison/Readme.md ; for section 5.2.2,
refer to Articles/2020-IJGIS-Temporal/DatasetVisualization/Readme.md and for section 5.3, refer to Demos/Temporal-LyonMetropole/Readme.md.

References


Appendix: Types and Enumerations

"dataType"  
DateTime

"dataType"  
Matrix4x4
+ values: double [16]

"enumeration"  
RefineValue
ADD
REPLACE
List of figure captions

Figure 1. Example of the evolution of a building and two possible models based on states modelling and on states and transactions modelling.

Figure 2. Six basic types of transitions between geospatial objects (Renolen 1997).

Figure 3. **Gen3DCity**: a generic UML conceptual model for delivering 3D city models on the web, generalizing 3D Tiles and I3S standards.

Figure 4. **Gen4DCity**: A generic UML conceptual model for delivering time-evolving 3D city models on the web based on 3D Tiles and I3S standards. Entities from **Gen3DCity** (from Figure 3) appear in white, and entities for temporal information manifest in grey.

Figure 5. An example of aggregated transactions.

Figure 6. **Logic4DCity**: UML logical model for time-evolving 3D city models on the web based on extending 3D Tiles. A logical model of 3D Tiles, **Logic3DCity**, is proposed in white and the temporal extension is represented in gray. The recursive composition association of **Feature** (thick line) is a conceptual representation of the *batch_table_hierarchy* extension of 3D Tiles.

Figure 7. Description of the concepts for extending 3D Tiles as a UML conceptual model.

Figure 8. Animation time comparison of *DS-3DTiles* and *DS-3DTiles-Tmp*.

Figure 9. Temporal navigation comparison between 3D Tiles and 3D Tiles with temporal extension.

Figure 10. Example of visual representation of buildings known states (state i and state i+1) and of transactions between these states. **States** are in gray, **creations** in green, **modifications** in yellow and **demolitions** in red.
Figure 11. Screenshots showing the evolution of a district of the city of Lyon (in 2013 on the left side and in 2014 on the right side) visualized in UD-Viz and using 3D Tiles extended with 3DTiles\_temporal.

Figure 12. Screenshot of the visualization of the city of Lyon in 2014 in UD-Viz, using 3D Tiles extended with 3DTiles\_temporal.

**List of table titles**

Table 1. Comparison of CityGML, 3D Tiles and I3S according to our needs for delivering time-evolving 3D city models on the web.

Table 2. Comparison of a data set representing the buildings of the city of Lyon, France (48 km²) between 2009 and 2015, stored in three different ways: in CityGML (DS-CityGML), in 3D Tiles (DS-3DTiles) and in 3D Tiles extended with 3DTiles\_temporal (DS-3DTiles-Tmp).

Table 3. Time for loading and rendering DS-3DTiles and DS-3DTiles-Tmp.