RIS3D, a Referenced Information System in 3D

Bruno Dutailly, Jean-Chistophe Portais, Xavier Granier

RIS3D, a Referenced Information System in 3D

BRUNO DUTAILLY, Archeovision UMS3657, Centre National de la Recherche Scientifique, France
JEAN-CHRISTOPHE PORTAIS, Ministère de la culture, Direction Régionale des Affaires Culturelles Nouvelle-Aquitaine, France
XAVIER GRANIER, Institut d'Optique Graduate School/LP2N, France

3D Geographical Information Systems (GIS), Building Information Modeling (BIM), … These words are increasingly used for Cultural Heritage studies to refer to the necessity to connect heterogeneous data with a 3D model.

In this paper, we introduce and formalize the notion of "Referenced Information System in 3D" - RIS3D, which extends the concept of GIS to 3D supports and generalizes it. For this purpose, we propose a generic and modular client-server approach that brings the required flexibility, extendibility and sustainability to deal with heterogeneous data. For queries and their visualization in a 3D environment, we push forwards the notions of layers and markers.

We also provide the readers with our technical choices that allowed us to develop our own implementation of a RIS3D solution. We demonstrate it on a real-case scenario: the preservation of the Lascaux painted cave where our solution has been deployed and is in use.

The proposed notion and generic architecture of RIS3D, its implementation and its deployment show the suitability of such an approach and its potential applications.

CCS Concepts: • **Computing methodologies** → **Graphics systems and interfaces**; *Virtual reality*; • **Human-centered computing** → *Visualization techniques*; *Visualization systems and tools*; • **Applied computing** → **Arts and humanities**.

Additional Key Words and Phrases: Computer vision, digital humanities, data visualization

ACM Reference Format:
Bruno Dutailly, Jean-Christophe Portais, and Xavier Granier. 2022. RIS3D, a Referenced Information System in 3D. *ACM J. Comput. Cult. Herit.* 1, 1, Article 1 (January 2022), 20 pages. https://doi.org/10.1145/3517043

1 CONTEXT AND CONTRIBUTIONS

Digitization of Cultural Heritage is now part of the standard tools in the field of archaeology. However, digitization alone is not sufficient. Documenting the corresponding artifact and site is the key to study and understand it. Documentation according to the diversity of disciplines involved is quite varied and has been adapted to new technologies. It is now a set by a huge amount of heterogeneous data. Indeed, an artifact or a site can be studied by archaeologists, architects, physicists, biologists, anthropologists, historians, chemists, geologists, … Each of them has their own data, expertise, analysis approach, acquisition systems with the same goal: to understand the studied object. Those data are organized and stored in different ways, but rarely in the same database or using the same ontology. 2D GIS is a common way to gather and organize everything in a common spatial referential, but is not adapted to 3D data, neither to insert it, nor to query or display it. Moreover, some specific fields such as caves need a real 3D support since layers and overhead view are not enough to grasp engravings or paintings on

Authors' addresses: Bruno Dutailly, bruno.dutailly@u-bordeaux.fr, Archeovision UMS3657, Centre National de la Recherche Scientifique, Archéopôle, Esplanade des Antilles, Pessac, France, 33600; Jean-Christophe Portais, jean-christophe.portais@culture.gouv.fr, Ministère de la culture, Direction Régionale des Affaires Culturelles Nouvelle-Aquitaine, 54, rue Magendie, Bordeaux, France, 33074; Xavier Granier, xavier.granier@institutoptique.fr, Institut d'Optique Graduate School/LP2N, LP2N-IOA, rue François Mitterand, Talence, France, 33400.

XXXX-XXXX/2022/1-ART1 $15.00
https://doi.org/10.1145/3517043

walls or ceilings. In recent years, many papers have proposed dedicated solutions to address thos problems (cf. Section 2): Geographical Information Systems (GIS) 3D, use of Building Information Modeling (BIM) ...

Inspired by the GIS definition where data are geo-referenced we introduce in this paper the notion of "Referenced Information System in 3D" - RIS3D (cf. Section 3) where data are 3D-referenced, formalized by a proposed generic architecture and some definitions. It shares similar features and concepts but it is not limited in the azimuthal 2D view. However, to make it possible to reference any data in a generic 3D environment, our approach thus proposes the following new features:

- a complete architecture composed of interoperable modules thanks to open protocols;
- data organization and typing that ensure the genericity and the upgradability of searchable information;
- solutions to ensure the independence between data and 3D support while preserving the cross-referencing;
- an extension of definitions from standards like "layers" and "queries";
- a visualization architecture to display and query data in a 3D environment.

We also propose a full implementation of those architecture and concepts (cf. Section 4).

Finally, we also demonstrate how our new software for storing, searching, and displaying scientific data in a 3D environment meets the user's needs (cf. Section 5). It is currently used by the scientific team in charge of the preservation of the Lascaux cave. We believe that our solution is not limited to heritage and can be scaled to any filed. This case study which already offers thousands of data from different fields is described and discussed.

2 PREVIOUS WORK

The most commonly known referenced information system is the 2D Geographic Information Systems (GIS). It has been used for decades [32] to record data with a geographical localization. In addition to classical database features, 2D GIS offers functions to organize records in layers, and makes it possible to perform queries based on 2D geometry, such as finding out elements inside a polygon or computing a distance. Those features are well adapted to archaeology since sites are excavated layer after layer, and elements are recorded accordingly. A lot of work and reviews have been published in this field with some of them leading to 3D wondering [21, 27, 35]. The first idea is to take advantage of existing 2D GIS and to add the support of 3D data or referencing [25, 26]. Then the 3D GIS appears and has been explored in a lot of dedicated solutions with either simple adaptations of 2DGIS or 3D models with associated database.

2.1 Moving GIS from 2D to 3D

3D GIS in the field of archaeology has been widely described as building information systems (BIM) and historical building information systems (HBIM) [15–17]. These works are well adapted to existing buildings, big amount of buildings, and virtual restoration of remains since they rely on BIM for urban managements such as GityGML [3]. In fact, for acquisitions made with laser scanner or photogrammetry, a process to convert them to buildings with their dedicated structuring is necessary to benefit from classical BIM. Such a specific structuring is hard to generalize to the diversity we can encounter in archaeology, from a single artifact up to a complex field like a cave.

Another approach is to segment 3D data into pieces homeomorphic to planes, and import them into classical 2D GIS [14, 31]. This workaround brings too many constraints on setting up 3D data for complex meshes like caves, but shows that it is justified to use a common database for heterogeneous data. Vertical dimension can be added, leading to what is commonly named as 2.5D models.

Other work focuses on extending 2D GIS features into 3D with the aim to provide a similar approach and workflow in the data management. As PostGIS [30] for extending PostgreSQL to 2D GIS features, SFCGAL [8] is used to perform 3D queries such as volume intersection, area and distance computation [24] and has been

integrated to GeoServer [34]. The SFCGAL project and its usage, shows how transposing 2D geometric operations to 3D is complex.

In conclusion, for these previous approaches, 3D in GIS are either simple 3D viewers, or vertical/height-field projection of more generic 3D data where multiple points cannot span the 3rd dimension.

2.2 Starting from 3D and add a database

On the other hand, fully 3D-based solutions start with semantic annotations based on 3DHop [10] or including the full process of 3D acquisition like Aïoli [28]. Those solutions are in general too closely associated to a given kind of object [10] or to a 3D reconstruction method (photogrammetry in the case of Aïoli). They do not offer the customization of data storage and are not able to query contents. 3D real time open source softwares offer to build new features to get close to 3D GIS [13, 19]. Those developments are specific to use cases and are not generic enough. Futhermore, the Barreau's perspective work [11] paves the way to a database displayable in 3D, but no implementation is proposed and the project is more suitable to virtual reality.

A generic information system referenced in 3D has been initiated within the research program "ArTaPOC" [20]. It consisted in the development of a 3D tool to free-handed crop a 3D mesh, and record metadata on it. 3D meshes and metadata were stored in a database, accessed by a WEB server through a WEB service, allowing to work together. The following "PHYT" [18] project has seen the developments of a more complex database to record analytical data in an experimental cave. Each discipline of the research team was able to create 3D anotations by picking landmarks on the 3D mesh, recording free texts and attaching any file. Quickly, scientists managed to define a common rule to formalize recorded texts in order to perform more accurate queries than looking for a word and then go beyond anotation. They also feedback the need to record analytical data directly in the database rather than in an attached file to perform queries on it. Those needs of a generic database and a finer-grained analysis of numeric data are parts of the guidelines that presided to the development of a RIS3D.

3 OVERVIEW

To overcome previous limitations, our proposed solution is a generic database able to be both queried and displayed in full 3D. This database implements the ontology [12] chosen by the user to fullfit their needs. It consists in a classical database without added limitation on data types (i.e., there is no need to define in advance what data will be inserted into), and a 3D visualization with query interactions. This genericity offers a new freedom to the user to define its own data type, and to query it. The visualization of such data is also a lock to be addressed.

Definition 3.1. A **referenced information system in 3D (RIS3D)** is a computer framework for storing, accessing, and displaying data related to positions on 3D models.

It is related to geographical information systems but adapted to the specificities of 3D models as well as the required extended genericity due to the multiplicity of potential objects that can be addressed. Similarly to text annotation [29], we use the same definition for "anchors" and "marker", adapted to 3D, and call "user's data" what is called "body" for a text annotation.

It is worth to notice that such a concept extends (and is thus compatible with) existing solutions that can be considered as partial implementations of a RIS3D.

3.1 Suggested Architecture

Seeking for modularity as shown in figure 1, our proposed architecture is composed of four main modules. First, we choose to propose a classical database engine with standard query language. Since modern database engine supports tree of graph representation and generic data storage, we consider that any ontology on data can be addressed this way. The 3D visualization is a separate module able to discuss with the database. A module between

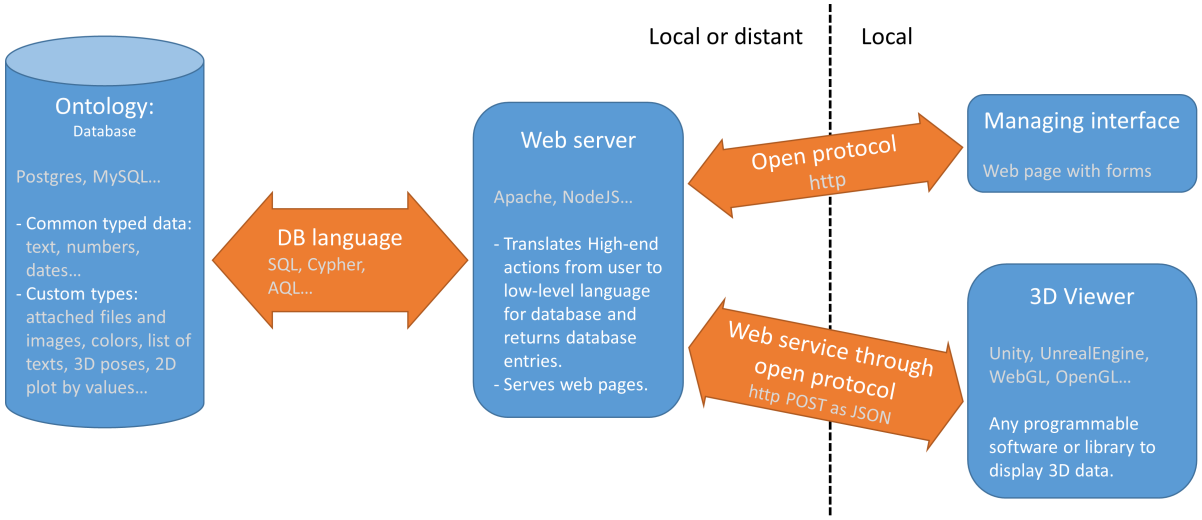


Fig. 1. Our proposed generic architecture of a RIS3D built with separated modules and open protocols. Grey texts are examples for implementation.

these two elements like a web server can ensure the communication and give an access to the managing interface module without a 3D support. It also facilitates the management of access rights and security controls with the supports of standards protocols in the field. The couple database and web server can be distant, if needed, and is able to accommodate multiple viewers and access at once. The communication language and protocol between the web server and the 3D viewer are open and allow to replace any module easily. Non-3D features can be accessed as web pages through the managing interface module.

3.2 Database module

3.2.1 Type handling. Setting a type to any stored data will facilitate its manipulation through dedicated features like inserting a new one, editing, displaying. Any type handled by a RIS3D must be storable in a classical database, compatible with queries, and displayed in a 3D viewer. Accordingly to users' needs and to existing database storage systems, a RIS3D is supposed to handle numbers, texts, booleans, and dates. This list is extended with images, documents, list of texts, and colors that are less common but needed. Following users' needs, new types can be added since they can be stored in the database using a combination of existing types. Those fields are parsed accordingly to their types to get full featured typed data. Then sensors or samplings could implement 2D graph as list of values, allow queries and show it as graph. Once defined, these types are used to store data and to give a handy access to it through dedicated interface and operator.

3.2.2 Data organization. Once the types are defined, data are organized in a graph or tree structure. This structured organization is needed by users to deal with the big amount of data that comes from different fields. In classical relational database engine, this is done by adding new tables and link them together with keys. In recent database engines, the JSON data type [22] can handle the tree structure natively. Moreover, the tree organization is already well accepted by users since they use it daily to store their files on their computers in the hierarchical structure of folders. Spatial analysis can be done by the chosen database engine. This could be PostGIS or SFCGAL. The web server can handle this as well on client side, thanks to libraries like treeJS [4]

3.2.3 Anchors: objects supporting data. All this custom data must be supported by an object in 3D we call an anchor. The most simple anchor is a 3D point defined by (X,Y,Z) coordinates. However there are other ways to anchor data in 3D.

Definition 3.2. An **anchor** is an object in 3D hosting user's data. The RIS3D handle four different anchors in the four dimensions:

(1) **0D**: a 3D point defined by its (X,Y,Z) coordinates. This location is provided by the user, from an external source or by pointing it in the 3D viewer. If available, the normal can be stored to be used afterward while displaying this point.
(2) **1D**: a 3D segment or a 3D path. This is an ordered list of 3D points.
(3) **2D**: a surface. Defining and storing such data is a complex task. We choose to store the user gesture while drawing a shape defining a surface. The camera parameters and transformation are also stored. These most native information can be replayed to build a 3D object regarding the 3D viewer implementation and performance.
(4) **3D**: a volume. We distinguish 2 ways to define a volume.
    - a 3D manifold mesh. This can be used for 3D architectural data based on simple 3D primitive, with a small count of polygons.
    - a grid of voxels. Those data can be obtained by geomagnetic survey, computer tomography acquisition, …

From 0 to 3D anchors, their definitions are simple to complex data to store in a database. The storage of a 3D point is obvious, but the storage of 1D to 3D anchors has to be addressed in the implementation.

3.3 Web server

The web server ensures the compatibility with the database engine and the communication between the user and the database. It offers a web service to communicate with the 3D viewer and web pages for non-3D operations like users permissions, integrity checks, … It also secures communications through the web and controls if the user has rights access to the data.

3.4 3D Viewer

3.4.1 3D geometry as backdrop.

Definition 3.3. A **3D backdrop geometry** is one or multiple 3D representation of the real studied site or object as a backdrop in the 3D environment.

These representations can be multiple as they describe differently the same part with different resolution of geometry or texture, or different modality like mesh or point cloud. Thus, any kind of 3D representation of this site or object is suitable for a RIS3D. 3D textured meshes built from 3D digitization or computer assisted design are usual in the field. The quality of the backdrop geometry is at the user's discretion. However, precision and resolution are important to set anchors with accuracy.

This 3D geometry is separated from the database. The link between 3D backdrops geometry and the database is granted by anchors since both use the same coordinates system. It is up to the 3D viewer to deal with projection issues if any, and hidden transformation if needed. This separation ensures to be able to load different versions of the same 3D geometry without having to convert, rebuild, nor redo records in the database. This happens when there are several hypotheses of restitution, different level of detail for the same geometry, new data acquisition, or evolution over time.

3.4.2 Layers.

Definition 3.4. A **layer** is the association of a query and a display setup of the answer.

The layer term is similar to a classical 2D GIS, but in 3D, there is no natural notion of above and below since it is given by the third dimension, so layers are not sorted. As in classical database, a query is a list of conditions to be verified to accept a record in the answer.

3.4.3 Display. Since the user can store any kind of data, and any amount of data, a choice must be done while displaying each record returned by the answer.

Definition 3.5. A **display setup** of a layer is a marker and a set of data mapped from the query answer to the marker capabilities.

Definition 3.6. A **marker** is the way of displaying user's data on an anchor in 3D.

Showing anchors as 3D object depends on the implementation of a RIS3D. However whatever the dimension of the anchor -0 to 3D-, showing it in 3D is not trivial since the 3D backdrop geometry is neither related to data nor to anchor. This means that the marker can be hidden by the 3D backdrop geometry. A marker is not only a 3D object, but it also provides functions. Each function can have one argument, and is able to change the aspect of the marker.

Definition 3.7. A **marker display function** is a function that can change the aspect of the marker accordingly to a given typed argument. A marker can implement zero to as many function as needed.

The user can map values from the database to these functions through its arguments. That means that each record returned by the query can be displayed differently according to the data itself. For example, all temperature sensor car be displayed with a ball which size represents the temperature value. Then large balls show high temperature whereas small ones show low temperature. This function system requires having the same type from the database field to the function argument. For more flexibility, a RIS3D includes converters to change the type of data.

Definition 3.8. A **converter** is a function reading a typed data as input and returning this data modified or in another type. It can be parameterized with arguments.

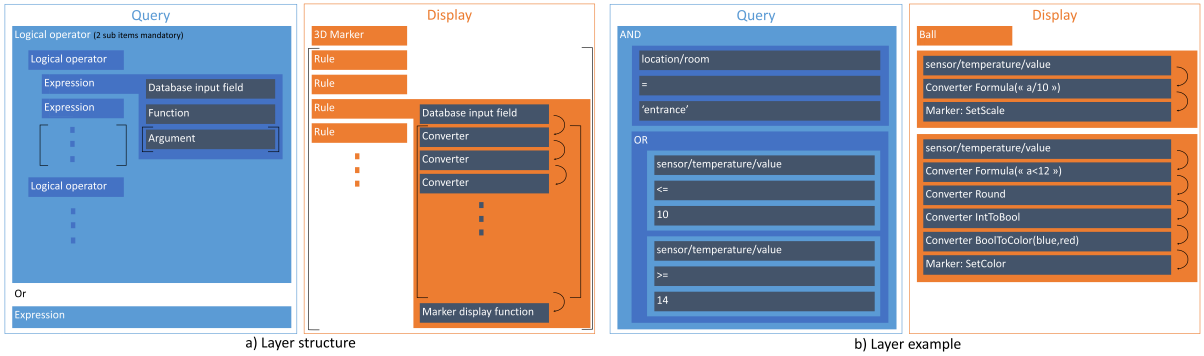
This conversion can be basic like changing an integer number to its text representation or parameterized like changing a floating-point number to its text representation with the precision given as argument. Thanks to this converter, a numeric value can be displayed in a text field of a marker.

Another simple converter is an algebraic formula evaluation to transform numeric values. Indeed, it is not required for a RIS3D to deal with unit of measure. If the 3D environment is defined in meter, 3D markers will be displayed in meter. The formula converter is then useful for transforming numeric values into more adapted values for display purpose.

Converters can be chained since the output of a converter is used as input of the next converter. The first converter will use a recorded data as input, and the output of the last converter will feed the marker display function.

Finally, mapping a database field value to a display function consists in picking a field address in the tree representation of data, optionally transform it through one or multiple converters, and feed a marker display function. An illustration of a layer's structure with converters and marker display functions is provided in Figure 2.

It is also possible to feed display function with static values not picked from database, but setup by the user. The same value will be used for all records displayed. For example, this is useful to scale all 3D markers to the same size without linking this appearance to a database field.



a) Layer structure b) Layer example

Fig. 2. Structure of a layer with converters and marker display functions detailed in a) and an example in b). Brackets shows optional elements. In the example, blue balls are the small ones with temperatures less than 10 degrees Celsius and red ones are the biggest with temperatures higher than 14. Temperatures between 10 and 14 are not selected by the query.

3.4.4 Data explorer. Once queried, data can be consulted and edited using a 2D user interface. It consists in a tree view showing the stored data, and providing tools to edit each element. As stated in the type handling section 3.2.1, each type defines how the data is displayed and edited. This 2D user interface can be displayed in 3D as a floating panel, attached to a marker, or in virtual reality, attached to a controller or to the player [37].

4 TECHNICAL DETAILS

For our RIS3D implementation, we have chosen a subset of features and we have done a number of technical choices (language, software, formats, database structure, web server, 3D viewer). In this section, such choices are explained and further detailed as an illustration of the generic definition of a RIS3D.

4.1 JSON

The JSON for JavaScript Object Notation [22] is a standard text format for data structured in a tree. It comes with couples of keys and values, where values can be a sub-tree, an array, a text, a boolean, a number, or a null value. We choose this format to describe user's data since any data can be stored as values organized in a tree structure. JSON is also used for user's preferences and personal data, and for communications between the web server and the 3D viewer.

4.2 Database structure

Most of database engines may be used according to RIS3D specifications defined in the previous section. We have chosen PostgreSQL [33] since, at that time and among other advantages, it offers one of the most adapted and effective JSON manipulation: JSON and binary JSON (JSONb) types and many functions to access and edit a part or a value inside a JSON record. Furthermore, it is compatible with the chosen web server. Finally, we use a unique JSON file to describe both an ontoloy (see section 5.3.1) and the database structure: evolution is thus easy to handle.

Mandatory data (i.e., non user defined data). They are stored in typed field. For 0D anchor and normal, the X, Y and Z coordinates are "double" fields. Two "timestamp" fields store the creation date and last modification date through default values or triggers functions integrated to the database. We discuss future possible implementation including database storage of 1 to 3D anchors in section 6.

User's data. They are stored in a JSONb field. The native JSON types are extended to a list of types meeting user's needs. Our full set of types are detailed in the supplemental materials section 1. The typed structure describing the tree organization of user's data with types for each field is defined as a JSONb record in a table "structure". Tree leaves values are all strings containing types of our RIS3D framework. More details can be found in the supplemental materials section 2. The user is free to build its own structure and fill it but the user's data are supposed to match this structure. If not, data will be ignored.

External documents. A table for external document is used to manage binary files stored in the web server, paths, names, and thumbnails for images. This table ensure the management of these files without making blob or base64 values in the JSON user's data. Links are made with a unique key referenced from the user's data.

User account. A table "account" stores user's id, login and encrypted passwords in fields, and a JSONb field contains preferences, user's personal data, layers, access control lists (ACL) and registered group of users.

4.3 Web server: the information multiplexer

Between the 3D viewer and the web server, all communications are transmitted in JSON. Between the web browser and the web server, web pages templates are sent in HTML, and data to feed them are sent in JSON.

NodeJS [7, 36] (v10.11.0) with the "express" [2] (v4.16.3) module is the web server engine of our RIS3D framework since its JavaScript language is perfectly adapted to JSON manipulation. A second key point is its ability to deal with many requests thanks to the non-blocking asynchronous system. In fact, the 3D viewer communicate with the web server sending frequently small requests, like performing a query to the database, updating a JSON field, downloading a document, saving settings, … Many other modules are used like `express-session` (v1.17.1) for login session, `bcrypt` (v4.0.1) for password encryption, `pg` (v7.5.0) for PostgreSQL interface, …

The non-3D features are supported by the web server through web pages in a web browser. Front-end technologies are JQuery (v3.4.1), JQueryUI (v1.12.1), Lodash (v4.17.11), and VueJS (v2.5.17). The list of non-3D features is given in supplemental materials, section 3.

4.3.1 Web service communication. A request to the web service is a JSON object that must contain at least the key "action" which value is a string defining the action. It may contain other keys that parameterize the chosen action. More details are provided in the supplemental document, section 4.

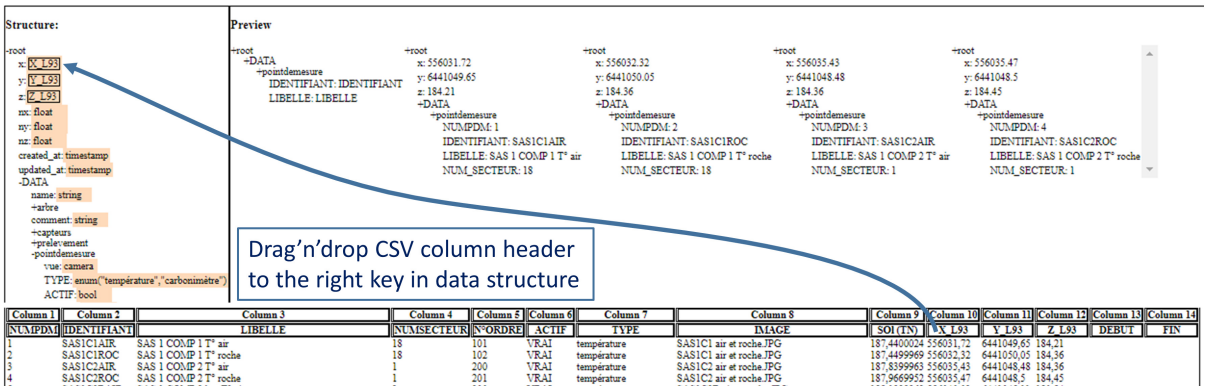
Once the request is performed on the web server, it returns a JSON object to the caller. This object contains an error code, an error message, and optionally a "response" key containing the data. The response data structure depends on the action requested. While performing a query to the database, the returned data is the set of matching records.

4.3.2 Setting up the data structure. The structure is a JSON object and can be easily edited by the JSONEditor [5] (v5.32.4) library and interface. It is also checked to ensure there is no mistakes like an unknown type set to a key.

At this stage, our RIS3D framework can handle complex operation to apply on data regarding modifications made to the structure. For example, renaming a key in the structure will lead to rename this key in all records using it. This is the same for removing a key. If the user remove a key, he is asked to remove the corresponding data.

4.3.3 Data import. Filling the database with user's data is a very important feature and CSV is commonly used for such an action. However, an important step consists in mapping the columns headers to the fields of the database structure. This is done manually through a web page where the user drags the columns heads to the unfolded tree structure as shown in figure 3. Real-time preview ensures that the mapping is correct.

Once the mapping is validated, the user can import all or a part of the input data at once. Since the type of each field is known, the mapping process includes a data conversion, like formatting dates, or converting text to



Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8	Column 9	Column 10	Column 11	Column 12	Column 13	Column 14
NUMPDM	IDENTIFIANT	LIBELLE	NUMSECTEUR	N°ORDRE	ACTIF	TYPE	IMAGE	SOI (TN)	X_L93	Y_L93	Z_L93	DEBUT	FIN
1	SAS1C1AIR	SAS 1 COMP 1 T° air	18	101	VRAI	température	SAS1C1 air et roche.JPG	187,4400024	556031,72	6441049,65	184,21		
2	SAS1C1ROC	SAS 1 COMP 1 T° roche	18	102	VRAI	température	SAS1C1 air et roche.JPG	187,4499969	556032,32	6441050,05	184,36		
3	SAS1C2AIR	SAS 1 COMP 2 T° air	1	200	VRAI	température	SAS1C2 air et roche.JPG	187,8399963	556035,43	6441048,48	184,36		
4	SAS1C2ROC	SAS 1 COMP 2 T° roche	1	201	VRAI	température	SAS1C2 air et roche.JPG	187,9669952	556035,47	6441048,5	184,45		

Fig. 3. CSV import: CSV column headers are drag'n'dropped to keys in the data structure and a preview shows in real-time the five firsts parsed lines organized in a tree following the structure.

boolean values for example. All this processing, until adding records to the database, are done in javascript in the front-end page. The CSV file is not uploaded to the server.

4.3.4 Exporting data. Data export can be done for the entire database, by table, or by layer. Export format are JSON, CSV and Excel. The library Excel4Node [1] (v1.7.2) has been used for excel export. For CSV and Excel formats, user's data are flattened to fill columns. The column header is the path to the field constructed with keys attached with slashes.

4.3.5 Database checks. The database structure is checked and updated if needed. This includes table fields and their types, triggers and functions. The user can also export database, drop tables, and import a table or the entire database. The user's data can be checked according to the typed structure. Our RIS3D framework is capable to list fields in user's data that are not declared in the structure, empty text values, text values which are supposed to be an element of a list declared in the structure, but which are not found in the list. It also checks all links between user's data and the table for external file management and thus provided list of unreferenced files and missing files.

4.4 3D viewer

We chose Unity as the 3D engine of our RIS3D framework. Note that all the database, communication and 2D/3D concepts for visualization and interactions are independent of this choice, and can be transposed to similar platforms. However, it allows fast development since it combines many features:

- it offers a very good 3D environment for loading complex 3D backdrop geometry, with textures, and even with levels of detail, if needed;
- applications can be built for numerous platform and devices;
- it includes communication with a web server through the unity C# WWW object;
- it comes with a user interface (UI) support, with simple widgets can be displayed on 2D screen or on 3D floating panels.

Furthermore, Unity supports the last rendering technologies like shaders, virtual, augmented and mixed reality. The scripting language is C# and is compiled during the build process, which is suitable for real time 3D .

The viewer in our framework is a Unity scene containing the user interface to be able to navigate into the 3D environment, to discuss with the web server and to display data in 3D. Backdrop geometries are added afterward as "addressable assets", built in a separate Unity scene. Our implementation of a RIS3D also uses the asset "new ui

widgets" [6] available on the asset store. This asset adds sophisticated widgets like list and tree views, colors and date pickers, and layouts. Virtual reality is available thanks to SteamVR plugin [9] for Unity.

Finally, JSON manipulation uses C# Linq library. Translations for internationalization are stored in a JSON file per language and loaded dynamically at startup.

4.4.1 Navigation. All navigation controls [23] are possible. We have implemented "rotate around" and "fly" modes.

The "rotate around" mode has a target located in the 3D environment: the view always look at, and rotate around it. To avoid up-side-down configuration, rotation are limited to zenith and nadir views. With this mode, the user has access to the mouse pointer and is able to set the target position to a point of a backdrop geometry, to create new record, to select displayed data, and to access to the 2D user interface.

A minimap is available and can display rasters provided by the user (see figure 7). It also displays active markers as dots. If a `setColor` function is used by a marker, the dot is colored with the color passed to this function. The minimap can handle positions and orientations. Those cameras are saved as the current view transformations in the user's settings and can be used to restore the current camera at this view.

4.4.2 3D backdrop geometries and other data to be imported. There is neither file reader nor importer in our RIS3D framework. In our Unity implementation, 3D backdrop geometries must be either setup in a fresh scene and exported as addressable objects with correct labeling. Labels are used to automatically import prefabs in the right way, for example, to distinguish a 3D backdrop geometry with a 2D raster for the minimap. The user can also label 3D object as invisible colliders for VR teleportation.

Setting up 3D data this way is useful to prepare the data, customizing textures, behavior with lights, levels of details, … The user is free to use the power of Unity. Only custom scripts are prohibited. Since all the code must be compiled with the main application, it is not possible to import code in addressable assets. The list of 3D backdrop geometries is accessible in the 2D interface and each can be turned on or off.

4.4.3 2D user interface. All 2D interface are setup in a "canvas". This canvas includes the minimap and is mainly composed by the layer panel. This panel offers the list of layers already available and a way to create new layers. Each layer can be unfolded to show the list of records answered by the query. Two buttons give access to two panels: one for setting up the query, and the other to choose and customize the appearance of the marker. A button force the query to be executed on the server and another force all markers to be redrawn. Other buttons are used to delete, duplicate, rename, or share the layer.

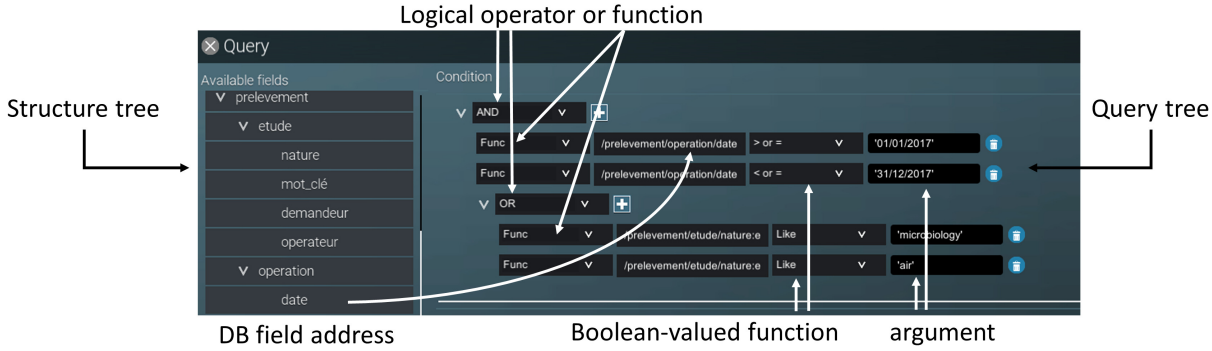


Fig. 4. User interface of a query with two functions and a logical `AND` between.

Query Interface. As illustrated in figure 4, the panel dedicated to edit the query is split in two parts. On one hand, the structure tree gives access to all elements available in the database. On the other hand, a query can be filled by logical operators `AND` and `OR`, or by a function. A function is constructed with a field that can be filled by the user with the help of completion or drag'n'dropped from the structure tree, an operator picked in a list, and the optional argument freely filled. The illustrated interface ensure there are a minimum of two sub elements to a logical operator, and a function has no sub element. The tree defining the query is serializable in a JSON string for storage purposes and for transmission to the web server.

The panel for customizing a marker's appearance is a bit more complex. First of all, a marker is chosen from a list. In our current implementation, we support three different 0D markers: the ball, the cube, and the billboard. The ball and the cube have a function to scale the geometry, and another to change the color. The billboard has a function to change the 2D icon displayed as a rotating sprite, a function to set the size of this icon, a function to set the height of the mast supporting the icon, a function to set the color of the mast and the icon, and three functions to set three texts around the icon.

Like in the query panel, a first part is dedicated to the available structure fields. This list is enlarged by the types defined in our framework in order to define, if needed, constant values rather than values recorded in the database. A second part is the list of rules to be applied to the marker. Drag'n'dropping a type or a structure field in the list creates a new rule. A fresh rule is composed by the input field or a typed defined in our framework and a marker display function. For these types, an input widget is used to set the constant value. This widget depends upon the type. The display function is a list of available display functions according to the input type given in the rule. A default value can be set on this display function and will be used if the input field is not found in the database record. Between these two elements, the user can insert several converters. Each converter has an input and an output type. The input type must be the same as output type of previous converter or constant type or structure field type if it is the first converter. The output type of each converter must be the same type as input type of next converter or marker display function input type if it is the last converter. If needed, arguments can be given to each converter. Those arguments are typed and widgets are provided for each argument to be filled. Some of provided converters are basics like changing a number to its text representation, or changing a text to a color if the text is well formatted. Others are more sophisticated and need to be parameterized with arguments. It is the case of the converter from the type "list of texts" to the type "color". In this converter, arguments are a color for each word of the list. With this converter, a marker can change its color to represent a text recorded as an element of a list. A list of converters is available in supplemental materials section 5.

Figure 5 shows a set of four rules, with constant and database field inputs, and the use of a converter with its arguments. Our RIS3D framework will check the complete chain of types and warn the user if there is a wrong type connection. The choice of the marker and all rules, including converters, are serializable in a JSON string for storage purposes.

4.4.4 Marker backstage. Markers are made in Unity as prefabs and implements an interface script to provide display functions. Since each marker is unique, scripts are unique too, so creating new markers needs a rebuild of our RIS3D framework. A marker is a 3D geometry that must fit in a one-unit cube. This rule ensures the scale functions to be uniform for all markers. A collider must be provided to make the marker clickable in our framework. In order to be interfaced with our implementation, a marker must implement a derived class and define an exposed function to communicate display functions and arguments to our RIS3D framework. The complete process is the following: for each record returned by the query, the chosen marker is instantiated, and all rules defined in the marker customization for this layer are applied, fed by records or constant values. Our implementation of a RIS3D comes up with several markers, and can be easily enhanced with new markers developed as plug-ins.

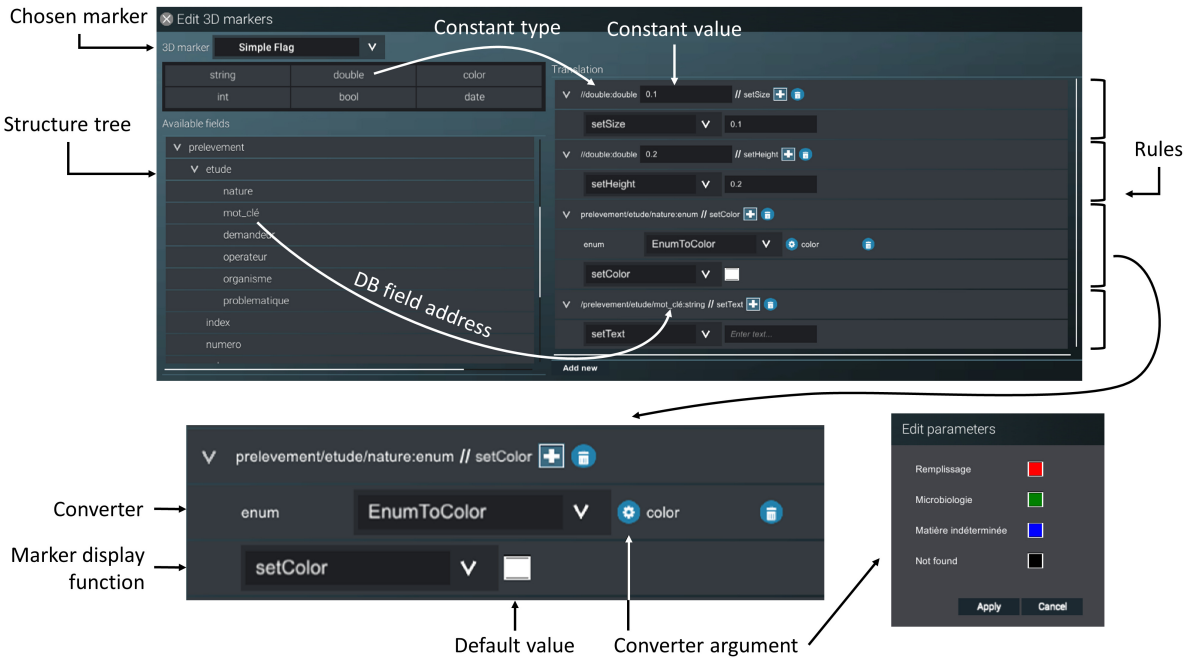


Fig. 5. User interface of marker parametrization. A rule with a converter is detailed at the bottom.

4.4.5 Converters backstage. A converter is a C# class derived from a mother class that defines name, input type, output type, arguments, and functions to apply the conversion. As for markers, creating a new converter implies to rebuild our RIS3D framework.

4.4.6 Access to user's data. With markers, user's data are visible in the 3D environment, and the aspect of the marker inform about the record contents. Some markers provide text displays in 3D, but it is difficult and obviously utopian to show all user's information since the amount is unlimited. This is the reason why a 2D panel is dedicated to display all these data in a tree view representing the data of one record. This panel is accessed by clicking on a marker or on a record in the response list of a layer.

Existing fields can be edited with a dedicated widget, regarding the associated typed structure (see supplemental materials section 6). A field can also be deleted. To create new fields, the structure tree is available and a drag'n'drop of a field will create the key in the tree view.

Each operation is sent to the web server to update the database accordingly.

4.4.7 Adaptations for Virtual Reality. Our RIS3D framework has been tested on a HTC VIVE VR headset. Simple and invisible colliders are used for teleportation. All 2D interfaces are displayed on a 3D panel attached to a controller. A laser pointer used by the other controller can target markers and interact with them. The same laser pointer simulate mouse click on the 2D interface panel, and all features are available in VR. By contrast, keyboard inputs are not supported. Some interactions are studied like speech to text or virtual keyboards to support data editing.

5 CASE STUDY: LASCAUX DECORATED CAVE

Disclaimer: Data presented in this section are only a part of the whole existing databases. This section shows a usage of Lascaux RIS3D, a dedicated implementation of our RIS3D framework and not scientific results about a study on Lascaux cave.

5.1 Presentation and needs

5.1.1 The Lascaux cave. The Lascaux Cave has been discovered by four teenagers on September the 12th, 1940, nearby Montignac village, France. It has been ranked "Monument Historique" on December the 27th, the same year. It became very quickly famous all over the world for its parietal wall paintings.

After huge work on accessibility, the cave is open to the public in 1948. The amount of visitors is constantly growing until 122,500 during the year 1962. But these visits and some morphological modifications (concrete planar floor, stairs) have weaken the preservation conditions of paintings. After the appearance of phenomena such as the white disease of the green disease,the French minister of culture, André Malraux, has decided to close the access to the public in 1963. Since 1972, the French Government is the owner of the cave. In 1979, it has been included in the UNSECO World Heritage List with fourteen other prehistoric sites and caves of the Vézère Valley.

5.1.2 Multidisciplinary monitoring. The scientific team in charge of monitoring the cave is composed by numerous specialists using different methodologies. They gather samples, measurements from sensors, observe the evolution of paintings, engravings and surfaces. Raw data are stored on a common server, but there are multiple databases on different engines, and it is time consuming to cross-check information between them. This tasks is made easier by a RIS3D which main goal is to put all data together in a common referential providing also a full access to all data to each user. A conversion process is an essential but hard task to take benefit from a RIS3D, and it depends on the input format.

5.2 3D inputs

5.2.1 3D backdrop geometries. The main backdrop geometry is a 3D mesh of the whole cave carried out by the Perazio Engineering company in 2014. It was acquired with a laser scanner for the geometry and hundreds of photos/photographs for texturing. The 3D model is segmented in smaller parts corresponding to the rooms of the cave. Each room is a separate mesh with multiple high-resolution textures (16k x 16k). This segmentation is visible in Lascaux RIS3D interface and each room is displayable independently.

The second is a meshed digital elevation map, also built by Perazio Engineering, for the land over the cave. It has no texture. An uniform green material has been applied for the display purpose. Trees have been virtually removed. They can be displayed as 0D markers since they are recorded in the database.

The figure 6 shows both backdrop geometries set up in a unity scene before exporting them to Lascaux RIS3D.

Other backdrop geometries have been imported, such as high resolution surface details issued from photogrammetry, or the fence of the parcel.

All backdrop geometries are setup in the same referential and unit, but can differ by their purposes, resolution, textures, or age.

5.2.2 Rasters. A raster map (see figure 7) representing the cave and the surrounding land in a nadir view is chosen as a minimap. Another version of this map with a shaded digital terrain from a LIDAR acquisition (created by Xavier Muth from Get In Situ) is also available.

5.3 Input typed data

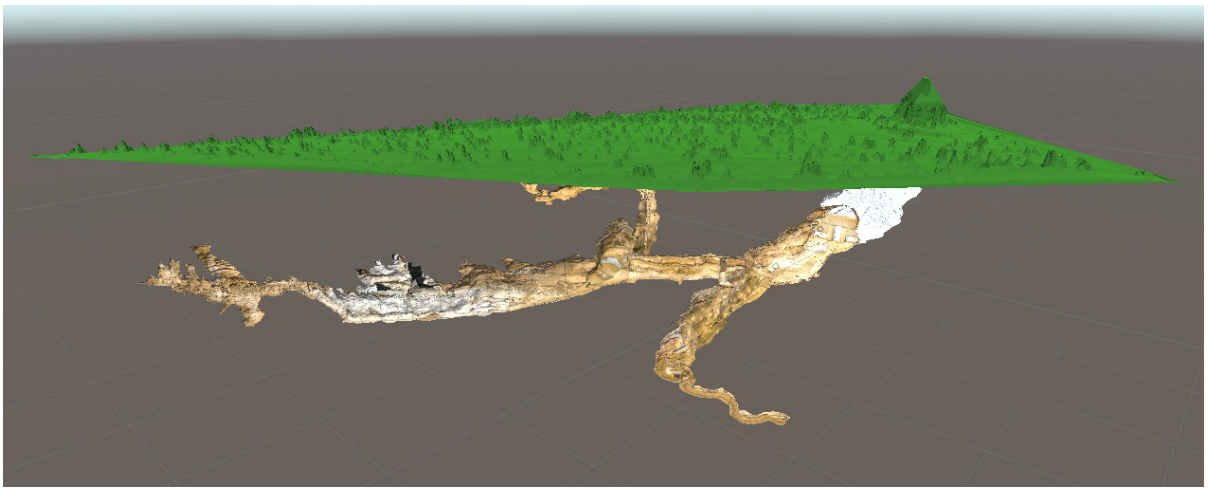


Fig. 6. Backdrop geometries for Lascaux RIS3D: the Lascaux cave and the above terrain (colored in green), set up in a Unity scene before the export into Lascaux RIS3D.

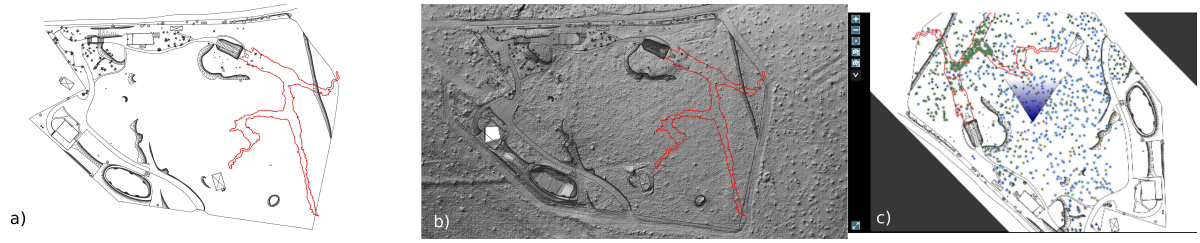


Fig. 7. (a and b) Rasters of Lascaux imported to Lascaux RIS3D for the minimap. (c) The minimap following the 3D view orientation and showing a raster and 2D markers.

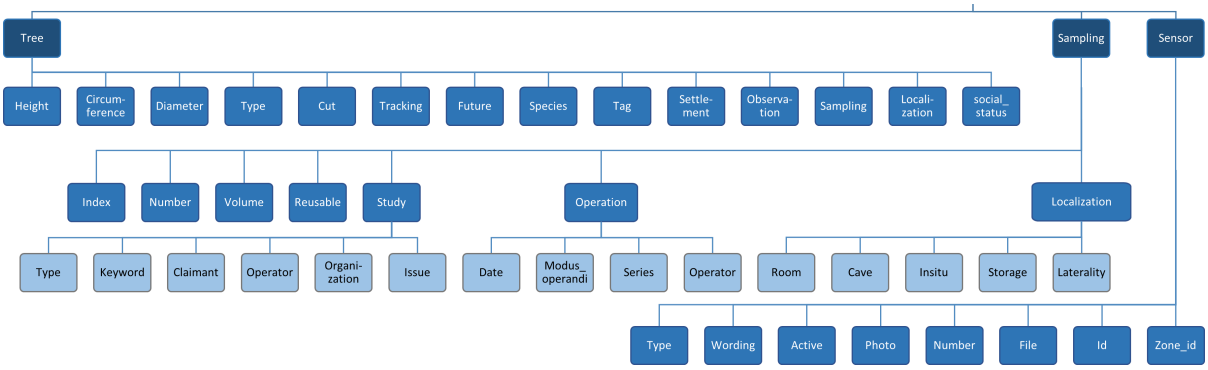


Fig. 8. Chosen ontology for Lascaux defined as a tree. Dark to light blue represents the depth in the tree.

5.3.1 Ontology. The ontology (see figure 8) has been defined by the scientific team and meets the different existing databases already in use. Naturally, the first level of the tree organization of the structure is dedicated to this dissociation. The team starts with three main sections: sensors, samplings, and trees.

Sensors are composed by: identification numbers, a type taken in the list of words such as "temperature" and "CO2" that will be extended to other types of sensors, a label as free text to describe the sensor, a boolean to state if the sensor is active, an image which is a picture of the sensor *in-situ* and, a file field to attach a file containing measures.

Samplings contain more information and some of them are ranked in subsections. At a root level, identification numbers and a volume can be found. Then a first subsection named "operation" describes who, when and how the sampling was made. A subsection "localization" is used to record where the sampling was done and where the sample is stored. And a last subsection "study" stores information about the type of sampling, who order it, from which organism, and the goal.

Trees has one level as sensors. It stores dimensions (height, diameter, circumference), species, type, observations, and identification tags.

5.3.2 CSV import from user's data. The database of Lascaux RIS3D have been fed with CSV exports from existing databases as these data were mostly previously stored in Excel documents. Importation of these data has consisted in defining a mapping between CSV columns and the structure defined in the previous section. This was done through the web interface of Lascaux RIS3D. It took few tens of minutes to map keys but, afterward, the importation of all data is very fast. More than thousands of trees, near 700 samplings, and 70 sensors have been imported that way. Note this is currently only a part of the whole existing databases.

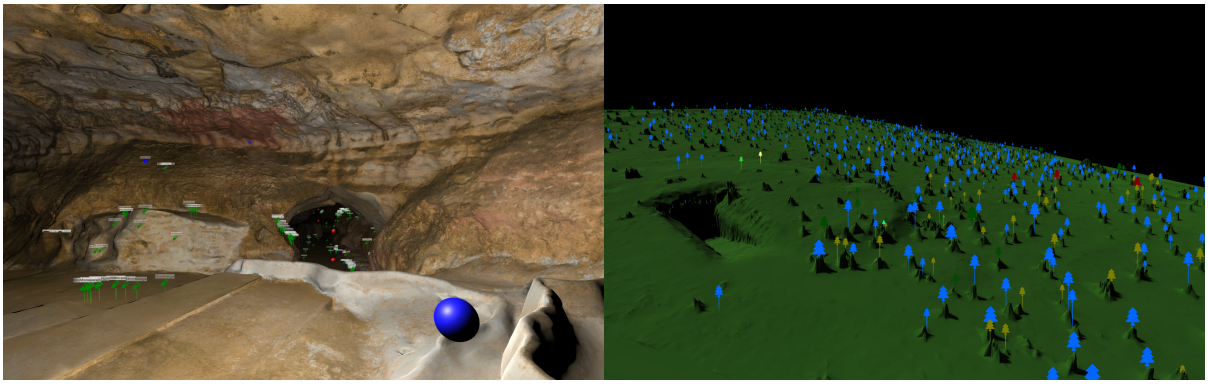


Fig. 9. Simple layers revealing sensors and samplings on the left, and trees on the right.

5.4 Layers

The first feature that takes advantage of a RIS3D is to simply data visualization in 3D. Since data stored in each record is heterogeneous, it must be filtered by simple queries. We have created three layers (e.g., figure 9) corresponding to the three first-level sections of the structure (cf. figure 8). Each query consists in checking the availability of a section in the data. Then, layers are displayed in 3D as 0D markers for each record filtered by the query. Each marker can have its own appearance defined accordingly to the data it represents. This is done by using the display configuration system of Lascaux RIS3D.

5.4.1 Sensors layer. We choose colored spheres for sensors, where the color represents the type of the sensor. Temperatures are red and CO_2 are blue. To achieve this, we map the data key `sensor/type` to the marker function `setColor`. Type of `sensor/type` is `enum` and `setColor` accepts only `color` as argument, so we add a converter from `enum` to `color` named `enumToColor`. This converter is configurable since Lascaux RIS3D detects the list

Fig. 10. Layer showing micro-biology samplings colored by room.

of words in the `enum`. The default markers' size (unit value, so 1 meter) is easily scaled using a constant value mapped to `setDiameter` function of a marker. In our case, we scale down to 0.2 meter.

5.4.2 Samplings layer. We have chosen icons as markers for samplings. The are chosen according to the field `sampling/study/nature`. Each icon is associated to a UNICODE and, once again, we use a converter to change the word taken from the `enum` to a `string` containing the UNICODE which is passed to the marker function `setIconFromUNICODE`. We also set some texts displayed around the icon, and as for sensors, we uniformly scale icons to acceptable values.

5.4.3 Trees layer. Like for samplings, trees are displayed using icons, but a unique one that is as a constant string mapped to the `setIconFromUNICODE` function. A color is assigned to each species contained in the data key `tree/specie` by converting `enum` to `color` as for sensors. The size of the icon itself is taken from the key `tree/diameter` and the height of the mast is taken from `tree/height`. Both are floating point numbers, but diameter is in centimeter in the database, and height is in meter. So for both settings, the `Expression` converter is set to do algebraic operation on the value, that consists in simply scaling it, and set better values for display purposes.

5.4.4 Advanced queries. As illustrated before, the customization of markers according to existing typed data available in a RIS3D leads to a natural and intelligible exploration of information: customizing markers following data in records is a good way to distinguish data variation in the same query. But we can go further with complex queries to more advanced filtering of data together with new tailored display customizations.

As an example, a layer displaying samplings made in 2017 is defined with a query composed with two conditions: field `sampling/operation/date` must be greater or equal to `01/01/2017` and the same field must be lesser or equal to `31/12/2017`. Note that the database engine return false in the absence of the requested field, so it not necessary to test its existence. In this layer, the display customization is the same as the samplings layer defined before.

As a second example, another layer showing all microbiology samplings is set with a condition on the field `sampling/study/nature` that must be equal to `Microbiology`. In this case, the color of the marker is changed according to the field `sampling/localization/room`. Indeed, the samplings layer had the color set to the nature, and in this new layer, all records are microbiology, and the color attribute visually reveals another information available in the records: the grouped localization as shown in figure 10. This view can reveal misplaced samplings, in their room record, or in their position in the 3D space.

5.5 Data view

Whatever conditions are in a query or display setup, all data of each record returned by a query are available in a tree view for reading, writing and deleting. Thanks to the typed database, each key has a dedicated widget according to its type to help the user to view and edit the value. The tree structure helps to find and set values, but it has been agreed to have formatted forms dedicated to subtree to ease the reading and recording.

5.6 VR support

A VR support has been tested during work sessions. While a user was moving in the cave usingVR device, another was on the desktop computer with the same view, and used the mouse as a laser pointer to point a region of interest. The VR and desktop view can be forecasted with a video projector allowing other users to discuss with the firsts.

5.7 Benefits

The use of the dedicated RIS3D by the scientific team of Lascaux is at the very beginning. It has already raised some errors of type in the existing databases, and some misplaced records without border effects on scientific interpretation of data so far. It thus already help in improving the database.

The 3D view of data has shown the distribution of samplings and strengthen the policy setup for up-coming sampling operations. It is also very suitable for discussions about sensors to be sure that every one are talking about the right sensor at the same time. The distribution of trees is also helpful for defining tree cutting strategy regarding species for the land preservation.

It is planed to feed the database with sensor values and query them to reveal uncommon behaviors like excessive temperatures or high CO_2 rates.

Beyond the actual setup, users are able to create their own data structures and layers, which is a major advantage to the monitoring team of Lascaux.

6 LIMITATIONS AND FUTURE WORK

The features we have deployed are based on the needs of the presented application. However, a RIS3D includes to others features that we have considered and we are working on.

3D queries. We have designed our RIS3D to be close to traditional 2D GIS with similar features. However the addition of a dimension drastically complicate all 2D operations like “is inside a polygon”, or “rasterize my data”. In fact, the equivalent of a 2D closed polygon in 3D is a manifold mesh. Testing if a 2D point is inside a 2D closed polygon is easy and can be done with several libraries, and moreover directly in database engine like the PostGIS

extension of PostgreSQL. Equivalents in 3D are still experimental like in SFCGAL. Even worse, testing if a 3D point is inside a manifold mesh is complex and CPU consuming.

For regular structure, the 3rd dimension also increases the required resources. As an example, the rasterization of data in 2D GIS consists in making a 2D image where data are discretized into pixels. In 3D, it will result in a 3D image, such as a voxel grid.

Cross table queries. The actual interface allows only queries in one table at once. So it is not possible yet to make a connection neither with another table nor with itself. Since the 0D anchor is the only table available so far, the first idea is to make a self-join to find 0D anchors relatively to another 0D anchor. This could be done to find all 0D anchors closed to another with a distance lesser or equal to a value. The data structure and database is already able to perform such query, but the user interface is not ready to for such an inclusion.

Anchors and markers. Regarding storage in the database, a 1D anchor is a list of 0D anchors, so it can be addressed by a table junction, or a custom JSON field. For 2D anchors, like surfaces, the work in progress is to store a JSON representation of all data needed to replay the user gesture of the clipping operation. This ensures the separation between user's data and 3D backdrop geometry. Finally, for a 3D anchor, 3D manifold mesh or a voxel grid can be used, and could be stored in a separate file or in a binary field of the database.

For each of these new anchors, besides storage, custom database operations, edition, and markers must be addressed.

New types of data. Since types are specifics to an implementation of a RIS3D, it can be freely extended with the condition the underlying data is serializable in JSON. This is what have been done with the type "camera" storing parameters in a sub-object.

We plan to add 2D plots as a series of 2D coordinates with some additional information about axis, titles, scales, … We also want to expand database operations to be able to query these series and get underlying information, like a specific value, the minimum, the maximum, … Displaying such complex data will be also a challenge. The main idea is to create an image of the plot and display it as a 2D sprite.

Another user need is to show small custom 3D assets as meshes and store it in the database. This could be done as a new type "mesh" referencing a file (e.g., PLY or OBJ), or serialized as a list of vertices and faces, eventually extended with UV maps and textures. Displaying this kind of data in 3D will be done through a marker which geometry is the stored mesh rather than its own geometry. Position, orientation and scale will need to pay attention and stored with the mesh. Applications are numerous, especially on sites that have brought artifacts digitized in 3D.

Dealing with huge amount of data. Retrieving records send all user's data of each through the network. This can be a huge amount of data at once. The idea is to send only fields asked in the `where` statement of the query and the first level of the tree structure to show at least a cross section of the data. Data used by the marker for the adaptation of its aspect have to be sent too. Then, small request in real time will be done on the data if the user asks for it in the data view feature. There is no lock with this improvement but we must tell the 3D application that JSON is incomplete and where it is incomplete. We also have to deal with merging JSON at each small request.

7 CONCLUSION

In this paper, we propose, formalize, implement and, test the notion of "Referenced Information System in 3D" - RIS3D as a tool to store, link, cross-query and display heterogeneous data around full 3D digital models of a Cultural Heritage artifact or site. Inspired from Geographical Information Systems, a RIS3D is based on a client-server architecture with interoperable modules. This modularity allows to choose different technologies

for the database, the web server and the software to display 3D in real time. The storage can be local through portable software or distant thanks to the web server interface.

For sustainability, our implementation relies on open protocols. For the same purpose, we propose a data organization and typing leading to genericity and scalability of searchable information. Typed structure ensure also both a general robustness and the ease of querying the database, adding and editing data, and displaying the data in the 3D environment.

Our RIS3D approach promotes the independence between data and 3D support. For the visualization, we have propose a dedicated architecture to display and query data in a 3D environment. Visual presentation is organized by the way of markers and layers.

Our implementation of a RIS3D has been tested on a concrete case: the preservation of Lascaux cave. It is now fully deployed and the scientific team is using it to track the evolution of all the measurements and observations. This case demonstrates that such an approach and our actual implementation can bring answers to existing needs and is already suitable for many use cases, especially in heritage preservation where many data are collected.

The application domain is certainly not limited to Cultural Heritage and there are opportunities that a RIS3D can be used in other fields. The current implementation developed for a given application scenario fits users' needs, but does not yet support the full set of potential features. However, the number of application scenarios are increasing quickly and we are already working of such improvements to reach the full genericity of a RIS3D.

ACKNOWLEDGMENTS

This study receive the financial support from the DRAC (Direction Régionale des Affaires Culturelles) of Nouvelle-Aquitaine, French ministry of Culture. We would like to thank previous financial support through research programs leading to this study: the Artapoc and PHYT projects. We would also to thank researchers and curators who use our implementation for their feedback and ideas. Future work will be supported by the French government in the framework of the University of Bordeaux's IdEx "Investments for the Future" program / GPR Human Past. Thanks to JOCCH reviewers for their objective remarks and advice.

REFERENCES

[1] [n.d.]. Excel4node is a full featured xlsx file generation library allowing for the creation of advanced Excel files. https://www.npmjs.com/package/excel4node. Accessed: 2020-11-09.

[2] [n.d.]. Fast, unopinionated, minimalist web framework for Node.js. https://expressjs.com/. Accessed: 2020-11-09.

[3] [n.d.]. GityGML: open data model and XML-based format for the storage and exchange of virtual 3D city models. https://www.ogc.org/standards/citygml. Accessed: 2020-09-28.

[4] [n.d.]. The JavaScript 3D Library. https://threejs.org. Accessed: 2020-11-09.

[5] [n.d.]. JSON Editor is a web-based tool to view, edit, format, and validate JSON. It has various modes such as a tree editor, a code editor, and a plain text editor. https://www.npmjs.com/package/jsoneditor. Accessed: 2020-11-09.

[6] [n.d.]. New UI Widgets is a set of widgets and UI interactions; it also includes a widgets generator for the custom data types and converter from default Text to TextMeshPro. https://assetstore.unity.com/packages/tools/gui/new-ui-widgets-27226. Accessed: 2020-11-09.

[7] [n.d.]. Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine. https://nodejs.org/. Accessed: 2020-11-09.

[8] [n.d.]. SFCGAL is a C++ wrapper library around CGAL with the aim of supporting ISO 19107:2013 and OGC Simple Features Access 1.2 for 3D operations. http://www.sfcgal.org/. Accessed: 2020-09-28.

[9] [n.d.]. Valve maintains this Unity plugin to smoothly interface with SteamVR. With SteamVR developers can target one API that all the popular PC VR headsets can connect to. https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647. Accessed: 2020-11-09.

[10] F. Ivan Apollonio, V. Basilissi, M. Callieri, M. Dellepiane, M. Gaiani, F. Ponchio, F. Rizzo, A. R. Rubino, R. Scopigno, et al. 2018. A 3D-centered information system for the documentation of a complex restoration intervention. *Journal of Cultural Heritage* 29 (2018), 89–99. https://doi.org/10.1016/j.culher.2017.07.010

[11] J.-B. Barreau. 2018. Exploration, analyse et gestion d'environnements archéologiques virtuels. In *Archéologie : imagerie numérique et 3D (Séminaire scientifique et technique de l'Inrap)*, Inrap (Ed.), Vol. 3. Sylvie Eusèbe and Théophane Nicolas and Valérie Gouranton and Ronan Gaugne, https://sstinrap.hypotheses.org/1299. https://doi.org/10.34692/yxcx-c180

[12] WN Borst. 1997. Construction of engineering ontologies for knowledge sharing and reuse. Technology. *Universiteit Twente. Retrieved from http://www. doc. utwente. nl/17864* (1997).
[13] L. Calori, C. Camporesi, M. Forte, and S. Pescarin. 2005. VR WebGIS: an OpenSource approach to 3D real-time landscape management. In *2º Congreso Internacional Ciudad y Territorio Virtual, Concepción, Chile, 12, 13 y 14 Octubre 2005*. Laboratorio de Estudios Urbanos, Universidad del Bío-Bío, 37–43.
[14] D. Marco Campanaro, G. Landeschi, N. Dell'Unto, and A.-M. Leander Touati. 2016. 3D GIS for cultural heritage restoration: A 'white box' workflow. *Journal of Cultural Heritage* 18 (2016), 321–332. https://doi.org/10.1016/j.culher.2015.09.006
[15] A. Chenaux, M. Murphy, S. Pavia, S. Fai, T. Molnar, J. Cahill, S. Lenihan, and A. Corns. 2019. a Review of 3d GIS for Use in Creating Virtual Historic Dublin. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* (2019).
[16] C. Dore and M. Murphy. 2012. Integration of Historic Building Information Modeling (HBIM) and 3D GIS for recording and managing cultural heritage sites. In *2012 18th International Conference on Virtual Systems and Multimedia*. IEEE, 369–376.
[17] Pierre Drap, Julien Seinturier, Marco Canciani, and Benjamin Garrett. 2003. A GIS tool box for Cultural Heritage. An application on Constantine, Algeria, historical center. In *Proceedings of the 4th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage*. 203–212.
[18] B. Dutailly, V. Feruglio, C. Ferrier, R. Chapoulie, B. Bousquet, L. Bassel, P. Mora, and D. Lacanette. 2019. Accéder en 3D aux données de terrain pluridisciplinaires-un outil pour l'étude des grottes ornées. In *Le réel et le virtuel*. https://hal.archives-ouvertes.fr/hal-02440440
[19] B Fanini, L Calori, D Ferdani, and S Pescarin. 2011. Interactive 3D landscapes on line. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 38, 5/W16 (2011). https://doi.org/10.5194/isprsarchives-XXXVIII-5-W16-453-2011
[20] V. Feruglio, B. Dutailly, M. Ballade, C. Bourdier, C. Ferrier, S. Konik, D. Lacanette, P. Mora, R. Vergnieux, and J. Jaubert. 2013. Un outil de relevés 3D partagé en ligne : premières applications pour l'art et la taphonomie des parois ornées de la grotte de Cussac (ArTaPOC / programme LaScArBx). In *Virtual retrospect 2013 (Archeovision)*, Vol. 6. Ausonius, Pessac, France, 49–54. https://hal.archives-ouvertes.fr/hal-01919004
[21] N. Herrmann. 2002. GIS applied to bioarchaeology: an example from the Rio Talgua caves in Northeast Honduras. *Journal of Cave and Karst Studies* 64, 1 (2002), 17–22.
[22] ISO/IEC 21778:2017 2017. *Information technology — The JSON data interchange syntax*. Standard. International Organization for Standardization, Geneva, CH.
[23] J. Jankowski and M. Hachet. 2013. A Survey of Interaction Techniques for Interactive 3D Environments. In *Eurographics 2013 - State of the Art Reports*. https://doi.org/10.2312/conf/EG2013/stars/065-093
[24] K. Jedlička. 2018. A comprehensive overview of a core of 3D GIS. In *7th International Conference on Cartography and GIS Proceedings. Sozopol. ISSN*. 1314–0604.
[25] M. Katsianis, S. Tsipidis, K. Kotsakis, and A. Kousoulakou. 2008. A 3D digital workflow for archaeological intra-site research using GIS. *Journal of Archaeological Science* 35, 3 (2008), 655–667. https://doi.org/10.1016/j.jas.2007.06.002
[26] G. Landeschi, J. Apel, V. Lundström, J. Storå, S. Lindgren, and N. Dell'Unto. 2019. Re-enacting the sequence: combined digital methods to study a prehistoric cave. *Archaeological and Anthropological Sciences* 11, 6 (2019), 2805–2819. https://doi.org/10.1007/s12520-018-0724-5
[27] G. Lock and J. Pouncett. 2017. Spatial thinking in archaeology: Is GIS the answer? *Journal of Archaeological Science* 84 (2017), 129–135. https://doi.org/10.1016/j.jas.2017.06.002
[28] A. Manuel, L. De Luca, and P. Véron. 2014. A Hybrid Approach for the Semantic Annotation of Spatially Oriented Images. *International Journal of Heritage in the Digital Era* 3, 2 (2014), 305–320. https://doi.org/10.1260/2047-4970.3.2.305
[29] C. C Marshall. 2009. Reading and writing the electronic book. *Synthesis lectures on information concepts, retrieval, and services* 1, 1 (2009), 1–185. https://doi.org/10.2200/S00215ED1V01Y200907ICR009
[30] R. Obe and L. Hsu. 2011. PostGIS in action. *GEOInformatics* 14, 8 (2011), 30.
[31] E. Ployon, B. Sadier, J.-J. Delannoy, S. Jaillet, J. Monney, E. Boche, and J.-M. Geneste. 2012. Le SIG comme outil fédérateur de recherche interdisciplinaire: application à la grotte Chauvet-Pont-d'Arc (Vallon-Pont-d'Arc, Ardèche, France). (2012).
[32] S. Quinn. 2016. *GIS Essentials*. CreateSpace Independent Publishing Platform, North Charleston, SC, USA.
[33] S. Riggs and G. Ciolli. 2018. *PostgreSQL 10 Administration Cookbook*. Pack Publishing.
[34] H.-G. Ryoo, S. Kim, J.-S. Kim, and K.-J. Li. 2017. Development of an extension of GeoServer for handling 3D spatial data. In *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*, Vol. 17. 6. https://doi.org/10.7275/R5ZK5DV5
[35] A. Scianna and B. Villa. 2011. GIS applications in archaeology. *Archeologia e Calcolatori* 22 (2011), 337–363.
[36] Stefan Tilkov and Steve Vinoski. 2010. Node.js: Using JavaScript to build high-performance network programs. *IEEE Internet Computing* 14, 6 (2010), 80–83. https://doi.org/10.1109/MIC.2010.145
[37] Y.-C. Tung and A. Hertzmann. 2018. *Comparing Spatial Interaction Modalities for 2 D-Widgets in Productivity Applications in Virtual Reality*. Technical Report UW-CSE-18-10-01. University of Washington.