From the Arctics to Antarctica - A multimodular visualisation of data

Jonathan Westin[1], Tristan Bridge[1], Matteo Tomasini[1]

[1] *Gothenburg Research Infrastructure in Digital Humanities, Göteborgs universitet, Renströmsgatan 6, Göteborg, 405 30, Sverige*

Abstract

This paper outlines the structure of Multimodal Map, a tool developed at GRIDH to access and visualise place-based datasets. The Multimodal Map frontend, which is developed with a Vue3 framework that fetches data from a backend built in Django, is arranged as a series of distinct and interconnected views that lets the user interact with the material at different scale and level of abstraction. To support the wide variety of formats the different projects need to handle, Multimodal Map makes use of both custom solutions and several open frameworks and libraries. These include Open Layers for the geographical visualisations, OpenSeadragon for IIIF-images, potree.js for point clouds, 3D Heritage Online Presenter (3DHOP) for meshes, and relight-viewer.js for RTI Photography.

Keywords

Research Infrastructure, User Interface, Data model

1. Introduction

The Gothenburg Research Infrastructure in Digital Humanities (GRIDH) have developed a package of user interface modules organised around a data model specifically aimed at spatio-temporal visualisations. The core package, which we refer to as Multimodal Map (henceforth MuM) was first fully developed for the project *Extended Rephotography* where the researchers needed both a system to register data collected in the Arctics and a tool through which to visualise the spatio-temporal relations in the material. The dataset consisted of glacier observations, historical and present photography and rephotography, measurements, and 360-degree video recordings. Since April 2023, MuM has been adapted and developed to accommodate the needs of several subsequent projects, including *Reading the Signs, Göteborgs Jubileum 1923, Etruscan Chamber Tombs, Sonora, Stokkastovan, The Inscriptions of Saint Sophia,* and *Built Cultural Heritage in Antarctica.* Through these projects, MuM has been expanded with capabilities to view, perform measurements on, and evaluate 3D data, explore reflectance transformation imaging (RTI), browse and filter visual galleries of datasets, and group and sort documentation according to date or type. Hence, rather than offering semantic annotation and structuring or processing of text and images, functions handled much better by mature tools such as Recogito (https://recogito.pelagios.org), MuM is defined by a "linear modularity", a semi-rigid structure that moves from the visual establishment of context to the exploration of digitisations through media-specific tools. The common denominator for the MuM projects is that the data, primarily visual in nature, is organised around an exact geographical position and a moment or event in time, which allows both for spatial exploration and chronological presentation and filtering. However, data-modelling informs the structure of a dataset by establishing hierarchies. In the MuM projects these hierarchies – and therefore the data model – are instigated by the concept of

Huminfra Conference 2024, Gothenburg, 10-11 January 2024.

jonathan.westin@lir.gu.se (J. Westin); tristan.bridge@lir.gu.se (T. Bridge); matteo.tomasini@lir.gu.se (M..Tomasini)

approaching the material at different scales, from abstract to concrete. Thus, we argue that MuM also offers a conceptual *method* for data registration that guides how a dataset is structured.

The present projects that make use of the MuM frontend and backend span a wide array of categories and geographies, from the Arctics to Antarctica, but all collect and make available datasets related to heritage that have been structured around the concept of place and scale through their adaption to MuM. These datasets include documentation of inscriptions from threatened environments and inscriptions in Ukraine, documentation of Swedish pipe organs, data collections about street signs in Rwanda, historical photographs from Gothenburg, immersive documentation of glaciers and the remains of the polar expeditions, and high-resolution point clouds of Etruscan chamber tombs in Italy and log houses on the Faroe Islands.

2. General description

The MuM frontend, which is developed with a Vue3 framework that fetches data from a backend built in Django, is arranged as a series of distinct and interconnected views that lets the user interact with the material at different scale and level of abstraction; *the map view* (A), *the gallery view* (B), *the place view* (C), and *the object view* (D).

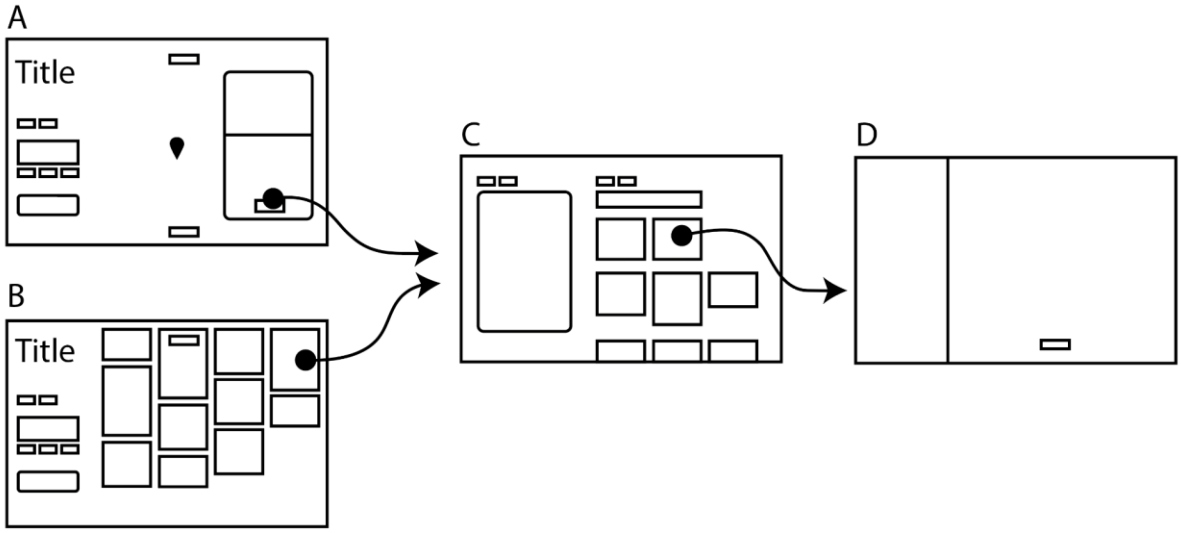

Figure 1: The conceptual connection between the different views. Illustration: J. Westin.

The main interface, *the map view*, is organised around the core component of a graphical representation of a Euclidean space where the data is spatially presented either as interactive markers or as overlays. Depending on the datasets that need to be visualised, this *map view* could be limited to the floor plans of one or several buildings (*Stokkastovan*, *The Inscriptions of Saint Sophia)* or be expanded to include an entire countryside or the better part of a continent (*Etruscan Chamber Tombs*, *Sonora*, *Extended Rephotography*). On the left hand of the screen, the user has access to a set of widgets through which to filter down the displayed data. These can be specified for each individual project and include controls that let the user limit the visible data to a particular dataset or period, view only data from a delimited geographical area or of a certain type, and focus on data that has been associated with a particular tag. As an example, in the *Etruscan Chamber Tombs* project, the user can filter by dataset, type of data (3d models, plans, or all data), time period (ranging from unknown to 400-200 BC), site, necropolis, and type of tomb. Such controls enable a level of data manipulation that is non-hierarchical through the ability to interact with specific sites as well as a more general filtering.

In parallel with the *map view* there is a *gallery view* which offers a graphical representation of the markers. Depending on the project, the gallery is populated with either a single visual representation of each of the places on the map (*Etruscan Chamber Tombs*, *Sonora*, *Stokkastovan*, and *The Inscriptions of Saint Sophia*), or all photographic data (*Extended Rephotography,*

Göteborgs Jubileum 1923). The map and the gallery mirror each other regarding what data sources they display, meaning that if the user filters down the dataset to a certain area on the map or a certain type of place, only images from that area or that type of place will be shown in the gallery.

Figure 2: The map view and gallery view of *Etruscan Chamber Tombs* with filter-widgets on the left.

When a marker is selected, data associated with that place is shown. Presently MuM has three possible interfaces for displaying this information; a place card that overlays the right hand side of the map and assembles available photos from the selected place into a carousel with a preview of the metadata presented below (*Etruscan Chamber Tombs*, *Sonora*, *Stokkastovan*, and *The Inscriptions of Saint Sophia*), a compact scrollable column that overlays the right hand side of the map with place data and previews of associated visual media (*Extended Rephotography* and *Göteborgs Jubileum 1923*), or as an expandable area to the right of the *map view* populated both with previews and metadata (*Reading the Signs*).

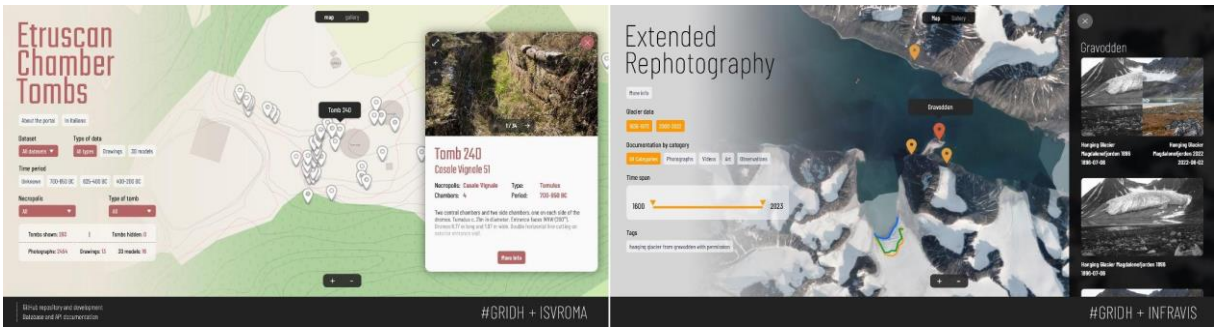

Figure 3: The place card in Etruscan Chamber Tombs and the column view from *Extended Rephotography*.

The projects that make use of the place card to preview the data all have an additional view, the *place view*, that the place card and images in the gallery link to. This view collects all the data from a place in an expanded interface that lets the user sort the associated data by type or date and presents a more generous space for descriptions and metadata connected with the place. In order to display and let the users interact with the wide variety of formats the different projects need to support, MuM makes use of both custom solutions and several open frameworks and libraries. These include Open Layers for the geographical visualisations, OpenSeadragon for IIIF-images, potree.js for point clouds, 3D Heritage Online Presenter (3DHOP) for meshes, and relight-viewer.js for RTI Photography. These libraries all come with their own user interfaces and have therefore been redesigned to present a coherent experience for the MuM user. When the user selects a preview image for visual data, the interaction is handed over to either the built in MuM *object view* for images, rephotography, and videos (both standard and 360) or for point clouds, meshes and RTI photographs to an auxiliary web-app built to handle that type of data.

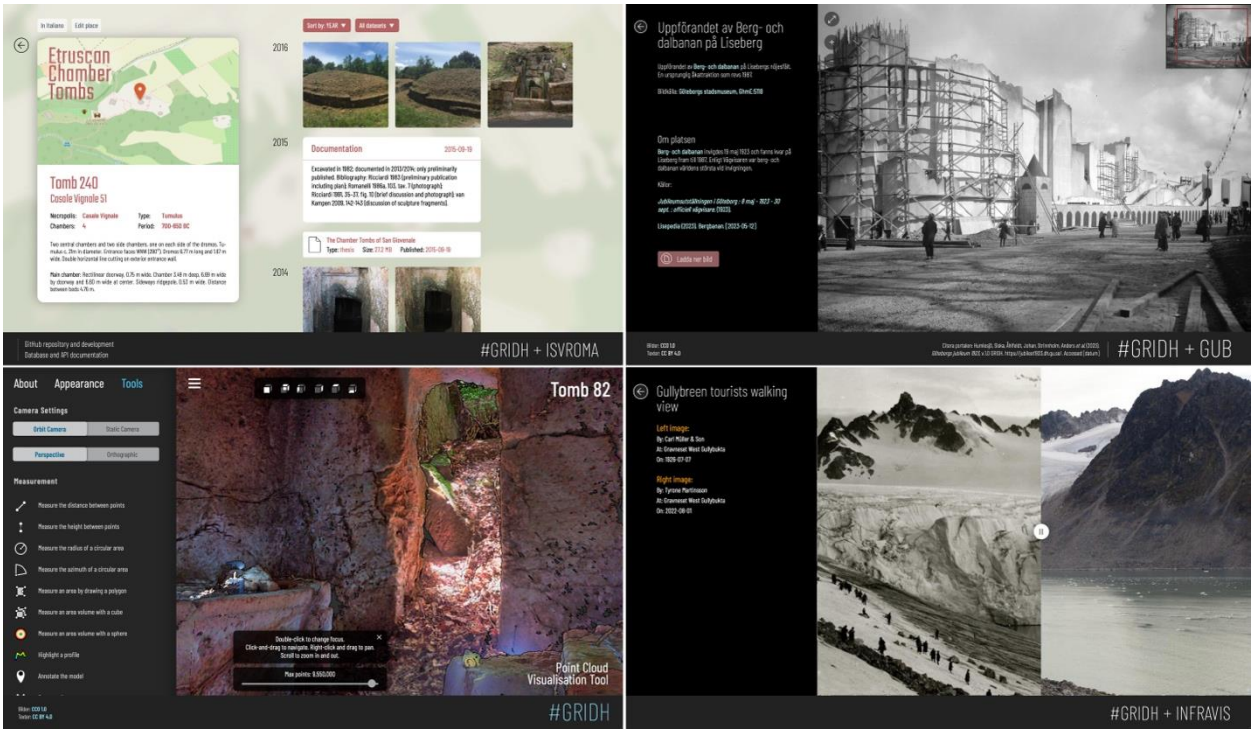

Figure 4: Place view and Point Cloud Viewer from Etruscan Chamber Tombs, Image viewer from Göteborgs Jubileum 1923, and Rephotography viewer from Extended Rephotography.

3. Frameworks and libraries used for MuM

In order to visualise meshes, point clouds, and RTI photography, the 3DHOP (https://3dhop.net), potree.js (https://github.com/potree/potree), and relight-viewer.js (https://vcg.isti.cnr.it/relight/) libraries were of particular interest. These come equipped with a wide range of features that facilitate user interaction and fast downloads of 3d assets through a pre-pyramidisation of the 3D data where only the necessary information is loaded at runtime. 3DHOP, for instance, provides built-in support for controlling scene lighting and carrying out measurements. Similarly, potree.js offers capabilities for camera manipulation and both distance, area, and volume measurement within a point cloud, and relight-viewer.js light manipulation and shader appearance. These features were considered invaluable for researchers interested in working with such material.

However, there are technical challenges involved in adding these libraries to MuM: 3DHOP and potree.js are predominantly jQuery-based libraries, while the main frontend framework for MuM is Vue3. The architecture and reactive databinding in Vue3 differ significantly from jQuery's more direct manipulation of the DOM (Document Object Model). Furthermore, Vue3 is designed to have primary oversight over its designated section of the webpage, and if jQuery makes any changes to that area, Vue3 may overwrite them during its next update. This leads to complexities in integration, as bringing jQuery into a Vue3 environment can result in conflicts and unexpected behaviour. Hence, a separate frontend/backend interface using Express and JavaScript were instead constructed. By building an auxiliary site with a near seamless connection with the main site, MuM-projects are able to utilise the 3DHOP, Potree and Relight libraries without the constraints of Vue3's architecture. This auxiliary site operates in parallel to MuM and serves as a unified platform for visualising and collecting all the point clouds and models across GRIDH's various platforms.

In addition to the auxiliary site built on the 3DHOP, potree.js, and relight-viewer.js libraries, MuM has currently support for streaming video and layered rephotography visualisation built on

open standards, and geographical representations and IIIF-image visualisation through OpenLayers (openlayers.org) and OpenSeadragon (openseadragon.github.io). Through its modular design, MuM can in the future easily be expanded with libraries and custom modules designed to access and visualise additional types of datasets as need arises, for instance through image clouds or WebXR, without breaking the overarching structure of the interface and the data model.

4. Diana and the data model

For its backend, MuM relies on a database coordination solution built by GRIDH, called Diana. Diana was written in Django with PostgreSQL, and it consists of an app providing base functionalities and abstract data models both for data input by the users and for making data accessible through generated REST APIs. With the tooling offered by Django, we can provide direct access to the database to our end-users, and limit their access to the projects they collaborate in. Through the Django admin site, the end-users can easily upload a variety of data without coding knowledge, and we can provide more tools for complex tasks such as batch uploads of data.

For each MuM project, a new application is written and installed in the Diana framework. Each application is generally centered around a data model indicating a Place, which includes some naming, categorization and, most important, a geography data field in the form either of geographical coordinates, or of a polygon indicating an area of interest. Places are then connected to other data models representing features such as images, 3D models, authors and/or reporters, observations, and various other forms of documentation, via Django's ForeignKey and ManyToMany fields (for more details, see Django Documentation). Tag models are used for categorization of other data models: in *Etruscan Chamber Tombs*, for example, we created tag-type models to describe different types of documentation, techniques used to develop 3D models, but also the epoch of datation of tombs. The data models for each project inherit some of their properties from the abstract data models provided through Diana. This ensures consistency in the database structure, while at the same time providing flexibility for the specificities of each project. For example, independently on what data models are specified within each specific application, each model comes with fields "created_at", "modified_at" and "published", that get automatically populated whenever a new data point is added to the database or modified. Some of the abstract models in Diana include *Tag models* and *Image Models*. *Tag models* consist of short case insensitive text, ideal for creating categories to which data points can be assigned. *Image models* include a field to upload IIIF-images (through GRIDH's IIIF server) as data points, as well as generating Universal Unique Identifiers automatically. This is but a short description of Diana's data models and how they interact with MuM, but a full treatment of Diana's capabilities is outside of the scope of this paper.

In addition to providing each MuM project with a database requiring minimal boilerplate code to be functional, Diana applications can potentially share data models and become an interactive powerhouse that gets more powerful the more projects make use of its functionalities. Diana shines also when it comes to the serialisation and generation of generic and consistent views in the form of REST APIs (including GeoJSON API), through the Django REST framework. This ensures the creation of compliant web APIs that the frontend relies upon. The flexibility of Django makes it easy to tailor these APIs to the needs of the frontend.

5. Conclusions

While it is close at hand to describe MuM as a *tool*, a software through which to register, access, and visualise a certain type of dataset, as has been shown it is to an equal amount a *method* for data organisation and curation; it informs an analytical approach to the material where a defined spatial position function as a fixture-point in Euclidean space for data of various types from various times. Hence, each point in space also becomes an archive of its own that organises data

pertaining to that place. The user approaches the dataset from an abstract representation of the data, as markers or analytical layers on a map or plan, but each step the user takes from there brings her closer to a more detailed representation of the data; first through a description of the spatially grounded place, and then through the individual representations of that spatial context served through the expanding set of visual data modules. The backend solution upon which MuM is developed allows for consistent data input and facilitates the interaction of end-users with the data shown in the frontend.

Acknowledgements

During the development of MuM, there have been several persons involved in providing code and solutions for both frontend and backend. Victor Wåhlstrand Skärström and Jonathan Westin instigated an early version of the frontend for *Reading the Signs*, then Arild Matsson, Tristan Bridge, and Jonathan Westin realised the first completed version for *Extended Rephotography* with backend support from Aram Karimi in Diana, which was initially developed by Victor Wåhlstrand Skärström. Tristan Bridge, Kristin Åkerlund and Jonathan Westin, with backend assistance from Johan Åhlfeldt, completed the *Göteborgs Jubileum 1923* and *Reading the Signs* iterations. Matteo Tomasini, together with Tristan Bridge and Jonathan Westin developed *Etruscan Chamber Tombs,* which served as a basis for *Sonora, The Inscriptions of Saint Sophia, Stokkastovan,* and *Built Cultural Heritage in Antarctica.*