University of Pisa
Department of Computer Science

Ph.D. Thesis: XXX cycle

Exploring Effective Publishing in the Web3D World

Marco Potenziani

Supervisor
Dr. Roberto Scopigno

Supervisor
Dr. Marco Callieri

April, 2018

Abstract

Web3D is most certainly an intriguing world. Its story has changed suddenly with the advent of WebGL, evolving from a slow and stagnant past to a dynamic and rapidly-evolving present. 3D data is becoming one of the key digital media on the Web, with a wide number of solutions specifically designed for publishing and consuming three-dimensional content online. Unfortunately, this field experimented a quick and often chaotic growth, presenting nowadays a dichotomy between pure research-oriented and market-oriented approaches. This has somehow shaped the directions of Web3D development, creating de-facto standards and solutions tailored to specific fields, or only focused towards mainstream publishing actions and thus unable to cope with the needs of more specialized and technical 3D users.

Under these premises, the aim of the thesis has been to investigate the shortcomings and missing features of Web3D technology, as well as to propose a solution aimed at filling these empty spots.

We start by presenting an analysis of the state of the art of 3D Web publishing, surveying the features provided by the major current approaches, useful to categorize the existing solutions and to cross-map them with the requirements of the different application domains. Then, in what is the main contribution of the thesis, we exploit the result of our analysis of the Web3D and discuss the design and implementation of a flexible platform, aimed at providing an effective framework for the Web presentation of specialized 3D content. Our solution is tailored to cope with the needs of a challenging application context, Cultural Heritage. Therefore it exploits highly-efficient solutions for data transfer and rendering, intuitive interaction/manipulation paradigms, and features enabling trans-media elements connections.

To validate the proposed framework, the thesis presents the results of two specific interactive visualization applications, addressing different Web3D presentation needs: a first one aimed at a museum dissemination initiative, and a second one developed to support scientific analysis.

Finally, we also tested the capabilities of our platform for the implementation of service-oriented applications: a project aimed at providing a service for the easy publication of complex, technical media types; and a more structured scenario of multimedia Digital Libraries, proposing a pipeline useful to rationalize and speed-up the publication of heterogeneous 3D dataset on a multimedia repository.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Our world is more and more overwhelmed by digital data. Among these information streams, the presence of 3D data is becoming quite common. Originally confined in the domain of Computer Graphics (the science of visually communicating via a computer display and its interaction devices), today 3D datasets are popular in every scientific discipline that relies on the power of modern computers to improve knowledge and speed up discoveries. Architecture, biomedical engineering, CAD design, fabrication, geo-exploration, computer vision, robotics, astrophysics, games and entertainment industry, cultural heritage, are just few examples of application fields leveraging on the analysis and processing of geometric 3D data.

However, the intrinsic complexity of 3D has characterized the exploitation of this informative layer since from its early days, limiting the chances to use it just to restricted tasks, like the basic visualization or the numerical analysis. Despite the efforts for going beyond these plain uses were relevant, for a long time a series of shortcomings (of functional interaction paradigms, or of tailored user interfaces, or even of smart modes for connecting 3D assets to other digital content), prevented the evolution of 3D data from a merely specialized support to a completely integrated content.

Nowadays these well known hurdles are becoming critical once again, since new trends in democratization of use and sharing of data are pushing 3D towards an unexplored world, where data management, user interactions, and cross-media integrations, are open issues still to be solved. Obviously the novel ecosystem we are talking about is the Web (democratic space "par excellence"), in recent years subject of renewed attentions concerning the integration of three dimensional content and the development of resources specifically aimed at this.

This thesis explores this rapidly-expanding environment, called *Web3D*, and it is aimed at proposing innovative solutions for the effective publishing of 3D data, after a careful study and analysis of the current status of the platforms, tools and libraries for the management of 3D content on the Web.

1.1 Web3D, Defining the Workspace

Generally, the term *Web3D* just means the presence of 3D content on the Web. Starting from this trivial formulation, it could be possible to affirm that every time a webpage embeds a 3D element, we would identify that as a Web3D implementation or instance.
However, from its inception, Web3D has become a more specific concept, and at the same time it has become the name of a community of developers/users, and the collective name for the wide ecosystem of algorithms, paradigms, libraries, tools, apps and services that manage 3D content online. When we refer to Web3D, we mean the higher-level idea of representing 3D models, scenes, animations, and interaction paradigms as a fully-integrated part of the Web environment. The emphasis goes on the concept of *data integration*, and on the strong interconnection between the 3D data and the Web paradigms, standards and base technologies.
Keeping this in mind, it easy to understand why for the purposes of this thesis, aspects like the possibilities to interconnect a 3D element with the DOM structure of the HTML page, the capabilities to create hyperlink-based connections between 3D content with other Web media, or the chance to fully exploit the HTTP protocol for the transmission of a 3D geometry, become predominant features compared to the simple presence of a 3D data online.

Our vision of Web3D is a context where 3D is not just a geometry that can be interactively rendered in a dark box on a webpage, but a media that should be fully integrated with the other existing ones already supported by the Web standards, and comply with the usual functionalities and methods of a Web publishing environment. Looking at Web3D from this new perspective, we can get a scenario certainly more defined, at least in its external boundaries. This, because, by looking at the inside of this workspace, we may find an indistinguishable continuum of publishing solutions, heterogeneous and rapidly growing in number. 3D on the Web is nowadays going through an expansion process, which should establish this data type as one of the key digital media for the Web. But the Web3D world has not always been so flourishing, and 3D content had to overcome several hurdles.

In early days, the Web environment was not designed to embed very complex data; consequently, it was not equipped with the necessary data structures for the 3D media. The lack of standardization resulted to be heavily penalizing for the first Web3D attempts, creating huge delay in the integration of 3D and the Web (especially if compared with the situation of other media types). Thus, for a long time, all the attempts to embed 3D content online only led to ad-hoc solutions, proprietary components, or external plug-ins, resulting in unsolved issues, limitations and incompatibility.

More recently, the impressive evolution of hardware, specialized Web software and network infrastructure capabilities, created the suitable conditions to turn 3D content into a standard Web media component. These technological advances, combined with a rising interest by the community for Web 3D applications, led to

the release of WebGL in 2009. WebGL is the standard API for Computer Graphics on the Web, whose introduction changed suddenly the status of Web3D by evolving from a slow and stagnant past to a dynamic and rapidly-evolving present. Thanks to WebGL, in a very short time-span, Web3D became an important research area, producing a number of relevant results. The renewed interest for Wed3D led to an important amount of related literature, proposing valuable approaches able to touch many applicative domains, but often resulting just in demonstrators rather than fully qualified solutions. However, it is important to note that Web3D received instantaneous attention also by software companies, which perceived it as a quasi-mature market already at the moment of the WebGL introduction. This generated instantaneous and sizable investments, resulting in a series of commercial-grade systems and software solutions, that constrain 3D content in a path very different from the one followed in the past by other kind of media, for which the Web exploitation followed a trial-and-error approach, with players of different size and wealth (from underdogs to large corporations). This market-oriented nature can also be seen in the fact that most of the available tools and systems do not have a reference scientific publication, even though they present innovative technical solutions.

Notwithstanding this dichotomy between commercial tools and academic experiences, the years immediately following the WebGL release saw the sudden availability of a number of usable systems that somehow shaped the directions of evolution and development of the Web3D field, creating de-facto standards (in terms of formats, interface paradigms, features, etc.), and solutions often only tailored to specific fields or to mainstream publishing actions, so generally unable to cope with the needs of more specialized and technical 3D data users.

All this has generated a Web3D panorama that, despite the wide variety of solutions available, at a closer inspection reveals several empty spots: unsolved issues, uncovered users and neglected fields. The presence of these empty spots has been the main motivation of this thesis.

1.2 Research Directions

Building a clear picture of the state of the art of the Web3D world is essential for discovering potential weak points and gaps in current methodologies, to identify possible open research areas and spaces for improvements. For this reason we decided to review in depth the status of Web3D systems and technologies, having in mind two main aims: firstly, to draw a characterization of the network interconnecting users requirements and available Web3D technologies; secondly, to derive, from the results of this review, the requirements needed for the design and the implementation of a software platform addressed to fill the possible empty spaces individuated in the Web3D panorama.
Therefore, we planned a systematic study of the available modalities for the

integration, interaction, and use of 3D content offered by the existing Web3D approaches and solutions, analyzing a heterogeneous set of software solutions as well as at the state of the art in Web3D literature. In planning this analysis (presented in detail in Chapter 2), we started from a list of ideal characteristics and specificities our review should satisfy:

- *flexible* enough to adapt to the extremely complex panorama resulting from the rapid growth of systems aimed at 3D Web publishing, in which a neat and rigid classification approach would be destined to fail;
- *representative*, in the sense of being able to give an exhaustive overview both of the software systems worth mentioning and the more remarkable academic papers concerning Web3D;
- *complete*, thus able to categorize the full Web3D landscape, going from low-level to high-level solutions, even though this means to survey and analyze a collection of contributions inherently heterogeneous composed by libraries, tools, frameworks, applications, etc;
- *multi-layer*, thus able to go beyond the classic "layer stack" structure and organization (where each software layer is built on top of another, and only vertical communication is possible), which seems to be unable to cope with the more fluid development of Web applications;
- *transversal*, in the sense of being able to cross-map the prominent and recurring Web3D peculiarities with the requirements of different application domains.

The final goal of our analysis is to define a reliable map that, unveiling the pursued "missing links", should be able to lead to the technical characteristics needed to build an optimal Web3D platform, given some application domains characterized by some degree of communality of scope and requirements.

By exploiting the results of this preliminary study, our next goals were: to propose a functional approach providing sensible answers to the listed needs; to design and develop a viable implementation; and, finally, to test this new solution over some concrete testbeds. Starting with this premise, the implementation of this platform should pose a strong emphasis on some key aspects, such as:

- *efficient* approaches for online *3D data representation/delivery*; eventually obtained adopting Web-oriented geometric data encoding formats and transmission/rendering approaches specifically designed for Web-based visualization;
- *context-based structural design*; eventually obtained adopting development paradigms oriented to the HTML declarative setup (familiar for developers

having a background in Web development), and a functional and modular structure derived from the idea to mimic the philosophy of other predefined HTML/JavaScript components available on-line (providing Web shaped event handlers, equipped with sensible default behavior, etc);

- *intuitive interaction/manipulation paradigms*; specialized on specific interaction tasks but at the same time customizable and interchangeable; possibly based on the exploitation of Web elements, and built to comply with heterogeneous application contexts and with different technical scenarios;

- *trans-media oriented connectivity*; eventually obtained adopting hyperlink-based integration schemes or others Web friendly components aimed to create multimedia Web pages interactively displaying 3D models;

- *multi-level* and *multi-target usability*; eventually obtained providing different configuration levels and modular components able to lead step by step to a fast and easy integration of 3D models into a Web page, and sufficiently easy-to-use (for developers without solid knowledge in CG programming), but also reusable in more complex contexts or in specialized environments.

Therefore, this novel platform should be an effective system, expressly designed for the on-line environment. It should be tailored on one of those specific niches of technical users which are still far from the mainstream use of 3D data and do not have a clear solution available for creating Web3D content.

The development of a solution based on this requirements resulted in the *3DHOP (3D Heritage Online Presenter)* system, a Web3D framework aimed at the interactive visualization of complex 3D dataset. This performing and flexible solution is presented in depth in Chapter 3.

Finally, to validate our proposal we planned to implement several testbeds, addressing different needs: firstly, to assess the platform consistency using it as a tool for Web content creation; secondly, to test the framework flexibility while using it for the design of complex services.

Concerning the first case, our goal was to contribute to the implementation of at least two real publishing experiences addressed to different communication aims: one presenting an example of Web publishing aimed at supporting dissemination in the framework of museum expositions, and another one developed for supporting scientific analysis and documentation. These two case-studies, both characterized by a design tailored on the requirements of the specific dataset (and so ideals to validate the modularity and the flexibility of our framework), are presented in detail in Chapter 4.

However, since these validation examples are mostly oriented to test the direct and static publishing task (i.e. the act of creating a complex visualization context for a specific 3D dataset), in Chapter 5 we will propose a couple of additional examples

aimed at exploring also the concept of service-based Web3D publishing. In this second case our goal was to assess the behavior of our platform firstly on supporting the easy automatic publication of complex visual assets on the Web (by designing a a 3D publishing service for "naive" users); and, secondly, on adapting 3DHOP technology to a digital library scenario, thus contributing to the implementation of an automated publishing pipeline aimed at rationalizing and speeding up the publication of large and heterogeneous 3D dataset in a specialized Web repository.

Chapter 2

Web3D as We Know It

The evolution of the technological solutions connected to the Web3D idea has gained a significant momentum in the last years, mainly driven by the introduction of the WebGL-factor. The resulting rapid growth of systems aimed at Web3D publishing is somehow changing the way to approach 3D data online, finally transforming them from merely specialized contents, used just by a small community of professionals, to a completely integrated media. On one hand this has contributed to familiarize users with the presence of 3D on the Web, on the other hand has created an extremely complex scenario, in which it is difficult to orient.

Nowadays, when we refer to Web3D solutions, we mean a wide group of approaches (from theoretical to practical endeavors, from low-level to high-level software, etc.) supporting Web publishing/presentation of interactive 3D content embedded into an HTML webpage. Considering that the success of a publishing action is not just related to the single media value, but strongly depends on the choice of the more suitable among those approaches, it is easy to understand how relevant is to be fully aware of the Web3D landscape. Moreover, having a clear picture of the state of the art of this world is also essential in discovering potential weak points and empty spots, so as to identify open research areas and opportunities for further work.

In order to cope with these needs, we present here a systematic study of the modalities of 3D content integration, interaction, and use offered by the existing Web3D approaches and solutions. Our main goal is to define a schema of the available possibilities, obtaining a map that, depending on the application field, could lead through the technical characteristics needed to build an effective Web3D presentation.

With this motivation we evaluated an heterogeneous set of software solutions and the state of the art of the literature, selecting and classifying noteworthy experiences and the more remarkable papers of the last thirty years of Web3D attempts. Nevertheless, the fluidity of the Web environment makes it difficult to outline a clear and rigid classification of the Web3D world denizens. To overcome this issue, we organized our analysis isolating some of the prominent and recurring

features that are used in the different publishing solutions, grouping them by their scope and functionality. Then, mapping these characterizing features on the existing systems we obtained a flexible schema able to present an exhaustive overview of the 3D Web publishing panorama.

We think that reviewing the existing through this analytical pattern could be very useful, not only in full understanding Web3D, but also to identify the current trends (in research and commercial development) and discover open challenges still to be faced in publishing 3D graphics on the Web.

2.1 Web3D, from Plugins to WebGL

Web and 3D professionals soon realize that 3D could not stay trapped in stand-alone applications and that opening the Web to 3D data would have had a great relevance. Already in 1994, a few months after the release of the first multi-media browser [SH94] able to mix just text and images, Raggett presented his vision for a platform independent 3D standard for the Web, VRML (Virtual Reality Modeling Language) [Rag94]. The Web3D denomination emerged immediately after.

Unfortunately, such a prompt start was not followed by the same pace in the development of practical and consistent solutions, and the path towards an effective Web3D resulted in a long and winding process. Some major pioneering landmarks were the Macromedia Flash [Cur00] plug-in (1996), the ancestor of Adobe Flash and probably the first approach to handle fully interactive multimedia content online; or the Apple Webkit CANVAS [Hic04] (2004), the first HTML drawing element controlled by means of JavaScript. Nevertheless, for a long time the Web landscape has been populated just by a series of proprietary systems, third-party software, and closed solutions. Not having a common and recognized development standard was a strong limiting factor for the real use of 3D on the Web.

The release of the WebGL API [Khr09] in 2009 was a major breakthrough, originating the rapid growth of a new generation of applications based on a common standard, able to act directly on the rendering pipeline and, above all, supported by all common Web browsers. Facilitated by the new standard, the proposed Web3D approaches start to specialize depending on the 3D content, the target users, the publishing venue typology, the application field, and the planned outcome. In short, thanks to WebGL, Web3D entered in a new era. But let us start from the beginning.

2.1.1 Early Approaches

The first attempts to publish 3D content online have led to a heterogeneous set of approaches, often really different from each other. This was mainly due to the lack of a real standard for 3D graphics on the Web. Actually, two different ISO standards solutions were available in the early phase: the already mentioned VRML [Rag94] and the X3D [Web04, BD07]. But unfortunately, since they were

designed basically as file/scene formats, they failed to cope with all the needs of Web publishing. Moreover they needed additional software to enable visualization in Web browsers. This last issue was probably the biggest hurdle to success of that generation of solutions. Those attempts indeed required the design (and use) of proprietary plugins, inaccessible to independent developers, and poorly integrated with the remaining webpage elements.

Beside the previously mentioned Adobe Flash [Cur00], the Sun Microsystem Java Applets [Boe09] probably constitute the earliest plug-in able to integrate computationally onerous content on the Web. Released in 1995, they are small applications executed client-side in the Java Virtual Machine, an abstract computing machine developed for running Java byte-code in a hardware-agnostic mode. Although the Java Applets probably constitute one of the first attempt to access the GPU from a Web browser (well before WebGL), they were not specifically aimed to manage 3D content. A follow up occurring in 1998 was the release by Sun of Java3D [Sun98], an API expressly designed to simplify the development of 3D Web applications. Java3D provided high-level constructs to create and manipulate 3D geometries, supporting both Direct3D and OpenGL. Java3D has been later discontinued, but, during the years, the effort to embed 3D content online using Java has been carried on by other libraries, like JOGL (Java OpenGL) [Jog04] and LWJGL (Lightweight Java Game Library) [LWJ07], known for being used in the development of the popular Minecraft [Per09] game.

The whole Java ecosystem has ignited and supported the design of interesting solutions at the beginning of the Web3D era, both concerning academic outcomes and software results. Just to cite a few of them: RAVE (Resource-Aware Visualization Environment) [GAW09], an applet designed for collaborative server-side visualization; COLLAVIZ [DDF+10], a framework for collaborative data analysis developed using JOGL; ParaViewWeb [JAG11], an applet for remote 3D processing and visualization (later adapted to WebGL); and finally OSM-3D [Zip10], an interactive 3D viewer for Open Street Map data developed as an applet.

The Java attempts have been one of the first approaches to support 3D content publishing online, but not the only ones. Indeed, just a year later Sun released Applets, Microsoft unveiled ActiveX [Mic96], essentially a framework for downloading multimedia content from the Web. Developed with a philosophy similar to Java Applets, it did not use files compiled in byte-code, but dynamically refers to OS libraries (which share with the browser the same memory space). This solution makes ActiveX very fast in execution, but causes drawbacks related to security and OS dependency which hindered a wider adoption of this solution. Microsoft changed approach in 2007, by releasing Silverlight [Mic07], a client side API for developing Web applications, this time closer to the Adobe Flash philosophy. Silverlight is another solution not specifically aimed to 3D, but able to provide a programmable graphics pipeline and basic 3D transforms, thus resulting a step forward from the Adobe Flash plug-in, which was not able to access GPU

functionalities until the release of the Stage 3D [Ado11] add-on.

Conversely to Adobe Flash, the Google O3D Project [Goo09] was designed since the beginning to support the use of GPU features (either via Direct3D or OpenGL). Released in 2009 as open-source plug-in for creating interactive 3D applications, this solution was ported to WebGL just few months after.

Finally, two approaches developed just before WebGL are the Opera Software plug-in [Joh07] and the Canvas 3D [Moz07] project (with its supporting library C3DL [Leu08]). These solutions, aimed to create an OpenGL context in the HTML CANVAS element, have been the real WebGL precursors, later becoming part of the new standard.

2.1.2 The WebGL Revolution

Anticipated by the aforementioned Canvas3D experiment, WebGL was finally announced by the Khronos Group in late 2009. Basically, it introduced a new standard for developing 3D applications on the Web which, in a very short period, reached an incredible diffusion, revolutionizing the world of Web3D.

WebGL is a royalty-free API fully integrated with the HTML DOM. It is based on OpenGL ES 2.0 [Khr03], the OpenGL API for embedded system (e.g. mobile or portable devices, possibly with lower-end computing resources, power consumption constraints, low bandwidth, reduced memory space, etc.). This makes WebGL extremely optimized and computationally light, thus ideal for the Web environment.

The main key advantages of WebGL are:

- to offer cross-browser and cross-platform compatibility. This means that a WebGL application can run on any platform without the need to re-write source code or install additional software.

- to be tightly integrated with HTML content. This includes layered compositing, interaction with other HTML elements, use of the standard HTML event handling mechanisms, etc.

- to offer a scripting environment that makes it easy to prototype interactive 3D graphics content. This means you don't need to compile and link before you can view and debug the rendered graphics.

- to be based on a familiar and widely accepted 3D graphics standard. This implies several advantages, like for instance the chance to use the GL shading language for programming the shaders (separate programs compiled at run-time and executed on the GPU).

- to give access to the GPU programmable pipeline, exploiting hardware-accelerated 3D graphics in the browser environment.

Thanks to these features, WebGL was able to checkmate the other 3D Web publishing approaches, contributing to the rapid extinction of all other solutions not WebGL-based.

Although it is designed to work strictly in conjunction with Web technologies (like HTML and JavaScript), WebGL remains a quite low level API, so it is not easy to master without solid skills in Computer Graphics (CG) and programming. For this reason, the years immediately after its release have seen the proliferation of middle-level wrapping libraries, aimed at making easier the use of this new standard.

Among these middleware solutions, one of the first to be released was SpiderGL [Vis10, DBPGS10, DBGB12], a JavaScript library providing typical structures and algorithms for real-time rendering. SpiderGL abstracts a lot of WebGL methods without forcing to use some specific paradigm (such as the scene graph, a hierarchical structure discussed more in detail in §2.4.2), neither preventing low-level access to the underlying WebGL graphics layer. Other mid-level libraries appeared almost simultaneously to SpiderGL were WebGLU [DeL10] (supporting a set of low-level utilities and a high-level engine for developing WebGL based applications) and GLGE [Bru10] (provides a declarative method for programming the 3D scene). Those were followed one year later by PhiloGL [Bel11] (a framework for data visualization, creative coding and game development), Lightgl.js [Wal11] (a low level wrapper, abstracts many code-intensive WebGL functionalities), and KickJS [NJ11] (a game-oriented engine, abstracts WebGL to make game programming easier).

Simultaneously with those mid-level JavaScript libraries, some focused solutions were also developed to build a bridge between WebGL and some popular 3D software applications. Among these: Inka3D [Wil11] (an exporter plug-in for Autodesk Maya [Aut98]), J3D [Dro11] (an utility to export static scenes from Unity3D [Uni05]), and KriWeb [Mal12] (a Blender [Ble95] exporter written in Dart).

2.2 Technical Background

The WebGL revolution opened a number of possibilities for using 3D on the Web, and this impressive potential has been clear right from the beginning. Nevertheless, while several characteristics of local 3D systems could be easily mapped on Web solutions, some technical arrangements were needed to handle with the peculiarities of the remote Web fruition.

In this section, we focus on a couple of main challenges that led the community to work on basic research in order to provide effective Web3D solutions. They concern: the dichotomy between *declarative* and *imperative* approach while defining an online 3D scene, and the management and the remote visualization of (complex) 3D dataset over a network. For each of these two challenges, we present a synthetic overview of the basic research and of seminal experiments that led to the current wide offer

of solutions, also providing a preliminary outline of Web3D base technicalities.

2.2.1 The Declarative/Imperative Dichotomy

The definition and handling of a 3D scene can be done with several levels of complexity, and the more complex the scene, the harder it is to provide a structure that can be easily used by non-experts. This issue is also true for local 3D rendering, but it poses severe challenges in the case of online 3D content creation and interaction.

A few years after the WebGL release, Jankowski et al. [JRS+13] presented a classification matrix that became popular in the academic world, being recalled in many later works ([JBG11, ERB+14]). Basically the proposed scheme introduced a parallel between the way to approach 2D and 3D graphics on the Web, classifying the available techniques into two main groups: *declarative* and *imperative*. Even if both of them allow to create, modify, share, and experience interactive 3D graphics on the Web, they differ on the basic approach and the target users. Indeed, while the *declarative* approach exploit the HTML Document Object Model (DOM) to provide access to high-level 3D objects (components familiar to the Web developers community), the *imperative* one uses scripting languages to offer access to low-level functionalities of the rendering stage (elements more common among the CG developers community).

Even though nowadays the distinction among the two approaches is becoming less and less substantial, the early years of Web3D were strongly marked by this dichotomy. This contrast was able not only to catalyze the attention of the research community (like [JRS+13] and similar works demonstrate), but also to influence a number of design choice which still characterize Web3D solutions.

Among the main causes of this separation there was the strenuous attempt to lead back CG solutions (and their developers) to the native Web environment approach, an effort bound to fail in the Web3D environment, a world in rapid evolution populated by solutions not confined in watertight compartments but rather inclined to overlap.

However, especially thanks to VRML [Rag94] (that represents a seminal work for the declarative Web3D approach), the declarative approach becomes dominant in many solutions between the late '90s and the early 2000s (ARCO [PWWS03, WWWC04, WCW06], ShareX3D [JFM+08], X3DMMS [ZCGC11]). VRML was essentially a logical markup format for non-proprietary independent platforms which used text fields to describe a simple 3D scene (only content and basic appearance). In 2004 VRML was superseded by X3D [Web04], in turn an ISO standard. X3D is defined as a royalty-free open standard and run-time architecture to represent and communicate interactive 3D scenes and objects using XML. It adds a lot of features to VRML (advanced scene graph, event handler, data-flow system for nodes interpolation, etc.), but analogously to VRML, it was designed basically as a file/scene format, and

therefore it needs an external software (e.g. FreeWRL [Ste98] or View3dscene [Kam04]) to be visualized in a Web browser.

Despite this evident limitation (and the absence of a continuous support), many solutions tried the way of the declarative approach in that first years. One of the first was the Blaxxun Contact plug-in (released in 1995, evolved in BS Contact [Bit02] in 2002). It is essentially a scalable multi-user server environment for the VRML developers community. Another pioneering declarative system was SimVRML [Ric02], that uses VRML for scientific simulations, followed a few later by other interesting attempts, like: Orbisnap [HUM05], a VRML97 viewer able to connect with remote servers running Matlab Simulink 3D Animation [MAT02]; Octaga Player [Oct06], a solution to build interactive 3D Web presentations; Cortona3D [Pac06], a VRML viewer and authoring tool (the last two solutions were later adapted to WebGL); and, finally, InstantReality [Fra09a], a VRML and X3D framework providing support for VR and AR.

In order to extend the browser support for X3D, X3DOM [Fra09b, BEJZ09, BJK+10] was proposed as a plug-in free declarative system, able to integrate the X3D nodes directly into the DOM content. Defining an XML name-space and using a special connector component, X3DOM builds a bridge between the DOM and X3D that allows manipulating the 3D content by only adding, removing, or changing DOM elements. An approach similar to X3DOM is the XML3D system [Son10, SKR+10, SSK+13], another plug-in free high level approach which, instead of embedding an existing framework (X3D) in the browser context (like X3DOM does), tries to extend HTML maximizing the use of existing features to embed the 3D content in the Web page. XML3D uses XHTML and, for intense online data processing (like for instance in [KRS+13]), relies on Xflow [KSR+12, KSRS13], an "extension" developed to expose system hardware and allowing data-flow programming.

X3DOM and XML3D have been the last two widespread solutions endorsing a "pure" declarative approach. Nowadays many of the features indicated as characterizing in the Jankowski's matrix mentioned at the beginning of this section (such as the scene graph construct for the declarative approach, or the WebGL API for the imperative approach) do not allow anymore to unequivocally classify existing 3D Web solutions. A brilliant demonstration of this is the A-Frame [Moz15] case, a solution released in 2015 that merges together a declarative structure and the famous WebGL supporting library Three.js [Cab10, Dir13].

Nevertheless, even if nowadays the declarative/imperative dichotomy is gradually losing meaning, it has been able to shape the development of Web3D over the years, and a full awareness of it can help to better understand many of the reference pattern of current 3D Web graphics systems.

2.2.2 Managing 3D Data Over the Internet

The issues related to the efficient visualization of 3D data over the Internet have long been both one of the most important research area and a characterizing topic of Web3D. The difficulties to interactively handling 3D content over a network mainly arise from these factors: insufficient computational resources (usually due to client/user side limitations), poor network capabilities (usually due to limited bandwidth and latency issues), and huge amount of data to process (usually due to the intrinsic complexity of the 3D content). As reported in the recent surveys of Shi and Hsu [SH15] and Mwalongo et al. [MKRE16] different approaches and techniques have been proposed over the time to solve these problems.

Web-based visualization in particular has received growing attention in the last years, with an increasing number of emerging applications. This success has been due to its ubiquity across platforms (from desktop computers to mobile devices), but mostly to the continuous growth of datasets and to the improvements in enabling technologies, such as server-side rendering infrastructures (via grid or cloud computing) or client-side rendering techniques (via WebGL and HTML5). Given the importance of interactivity in visualization, 3D Web publishing platforms have been mainly based on this latter rendering techniques, able to perform GPU-accelerated local rendering directly within modern browsers. Nevertheless, any of these improvements would solve the challenge of efficient visualization without efficient methods to manage and transfer 3D data between the server and the client. For that reason, techniques aimed to reduce bandwidth requirements (like compression and other optimizations, such as progressive rendering and streaming) have historically been very active research fields. This section tries to briefly cover some of these methods.

Nowadays all the main systems aimed to Web3D successfully exploit either *layered* (discrete sequence of data each of which represents the object at a different resolution) or *multi-resolution* (virtually continuous set of data representing an object at increasing resolutions) data encoding. Layered representation are usually called Level of Detail (LoD) representations [Lue03].

However, despite technology to performing handle 3D models has been studied since mid-90's [Hop96, PS97], decisive progress appeared only in the last decade, taking also into account the specific Web requirements.

In particular, when works like [LWS+13] start to prove the importance of balancing visualization latency and performing well in space/time, became clear the need to use all together technologies like LoD/multi-resolution representations, progressive data transmissions, and data compression schemes.

One of the first effort in this direction is the work by Gobbetti et al. [GMR+12], which proposes to transmit 3D models parametrized into a quad-based multi-resolution format. Lavouè et al. [LCD13, LCD14] suggested to iteratively simplify 3D meshes encoding their information in a compressed stream, thus adapting for the Internet the progressive algorithm of Lee et al. [LLD12], enabling

a low compression ratio with a small decompression time (solution based on the valence-driven progressive connectivity encoding introduced by Alliez and Desbrun [AD01]). Limper et al. [LJBA13] overcome the problem of decompression time/complexity using different quantization levels for the model vertices and transmitting them using a set of nested GPU-friendly buffers. More recently, Ponchio and Dellepiane [PD15, PD16] presented a performing multi-resolution rendering algorithm (parallelizable and scalable to very large models) based on [CGG+05] and able to combine progressive transmission of view-dependent representations and efficient geometric compression/decompression. Finally, to cope with different issues, alternatives directions have been proposed, focusing on improved data organizations (e.g. generic formats) [LTBF14, SSS14], or different data types (such as point clouds) [EAB14, Sch15] of streamable 3D content.

The works mentioned in this section do not only represent performance enhancements, but rather are real turning points for applications fields requiring to handle online very complex 3D content, where the capability to present data at full accuracy (without using degraded simplified models) is a major accomplishment. They have been able to change the recent history of 3D Web publishing, and, since 3D data compression and optimization of data streaming/rendering are still trend topics in research (the recent stable release of the glTF [Khr15a] asset delivery format is just a proof of that), surely they will be able also to deeply influence the next generation of 3D Web publishing solutions.

2.3 Feature-Based Characterization of Web3D Solutions

Since the WebGL release the resulting Web3D scenario growth in a myriad of different proposals, diversified depending on content, target, context, and of course, expected results. In order to define a schema of the available (or required) features, a categorization of these approaches would be both useful and needed.

2.3.1 Which Categorization of the Existing Solutions?

Chasing a representative categorization of the Web3D landscape, we have surveyed and analyzed a large set of solutions, starting with simple libraries, moving to middle-ware solutions, and up to complex applications. The decision to cover approaches going from low-level to high-level solutions (see Figure 2.1) has allowed us to cover almost entirely the 3D Web publishing possibilities and needs but, on the other hand, has returned a collection of contributions inherently heterogeneous, composed by libraries, tools, frameworks, applications, etc.

Even if the available solutions are characterized by different abstraction levels, they also reveal several deviations from the more classical level-based classifications. These contaminations often lead to fuzzy boundaries between the

Figure 2.1: A representation of the Web3D space reviewed, covering all the space in between low-level and high-level solutions.

various systems, making hard to reduce them to reference patterns. The idealized image of a clean "layer stack" application (where each software layer is built on top of another, and only vertical communication is possible) seems to be unable to cope with the more fluid development of Web applications. As most of the modern Web3D system are based on JavaScript, it is really easy to jump over layers, and interface at different levels to the various software components of the application. Many of what we would call "application layer" directly interface with the very bottom WebGL layer for some specific low-level function, but also exploit, at the same time, multiple "supporting layers", that may themselves have cross-dependencies. It is quite common for "WebGL wrapping libraries" to be designed in this way, that allows to use them as low-level libraries, but also as basic "load-and-display" applications. This fluid development makes sometimes difficult to categorize and restrict a software component in a single category, and the resulting scheme is something more complex than a neat series of software layers only communicating vertically.

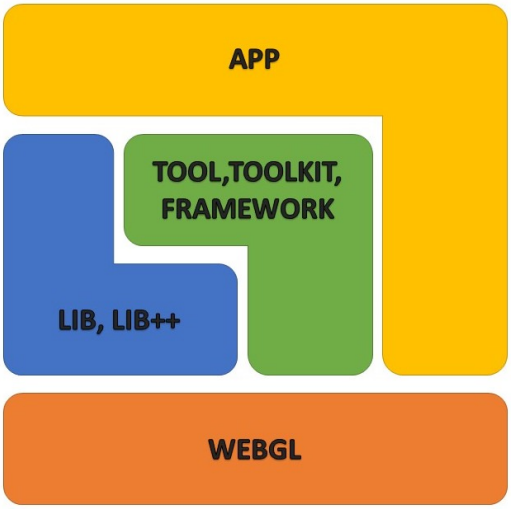

Figure 2.2: The diagram presents a layered representation of the software contributions used in Web3D implementations, which aims to define a map characterizing connections and relations among the different solutions.

The stratification proposed in Figure 2.2 tries to structure and better represent the scenario introduced in Figure 2.1 enclosing the attempts built over WebGL in three macro groups (defined not only by looking at the abstraction level of the solutions, but also considering other key factors, such as their interactions, the provided capabilities, coding needs, etc.):

LIB/LIB++ = Supporting libraries for the WebGL API (low level, coding is required). The notation covers the whole low level scenario, starting with basic libraries that do not provide high-level features, like Lightgl.js [Wal11] (bare-bone WebGL wrapper, abstracting many code-intensive functionalities), and gradually arriving up to more complete ones, like Frak Engine [3D 12b] (library providing features for simplifying the creation of complex interactive 3D applications), or Three.js [Cab10] (multi-level API allowing the developer to rely on a wide set of features, such as coding support utilities, documentation, examples, tutorials, how-tos, etc.);

TOOL/TOOLKIT/FRAMEWORK = Middle-level solutions. To access WebGL they often use one of the aforementioned libraries; usually provide a GUI; almost always require a certain degree of coding. May range from simple tools to complex frameworks. Besides the basic features, they can provide the developer with higher level functionalities related to the 3D scene (hot-spot, user interfaces, analytical tools, etc.). Examples: WebGLStudio [AEB13, Age13] (toolkit to create interactive 3D scenes directly from the browser); or Clara.io [Exo13, HLL+13] (platform for modeling, animating, and visualize 3D content online);

APP = Applications at the highest level, where authoring elements are used and coding is not needed. Typically end-products supporting the online 3D publishing in the form of Web services. Examples: 3D Wayfinder [3D 12a] (architectural application offers ways to manage content in 3D floor plans), or Pinshape [MSY13] (portal and marketplace for 3D printing community).

Although this layered scheme fits very well with a wide range of reviewed cases (in Figure 2.3, we present just three graphical examples describing some of the cited software), at a closer analysis, it turns out to be not flexible enough to provide a complete characterization of all the existing solutions. For instance, it fails to clearly classify "multi-level" libraries (wrapping libraries providing, at the same time, very low level and very high level access), or again to correctly identify solutions like 3D publishing standalone suites to download and install (end-products that act like middle-ware systems but at the same time provide very low level possibilities).

To accomplish our categorization we decided, therefore, to propose a different point of view: a systematic study on the *features* supported by the state of the art solutions, organized in a sub-set of macro-groups. The reason for following a *"per feature"* review rather than presenting the domain by listing and describing

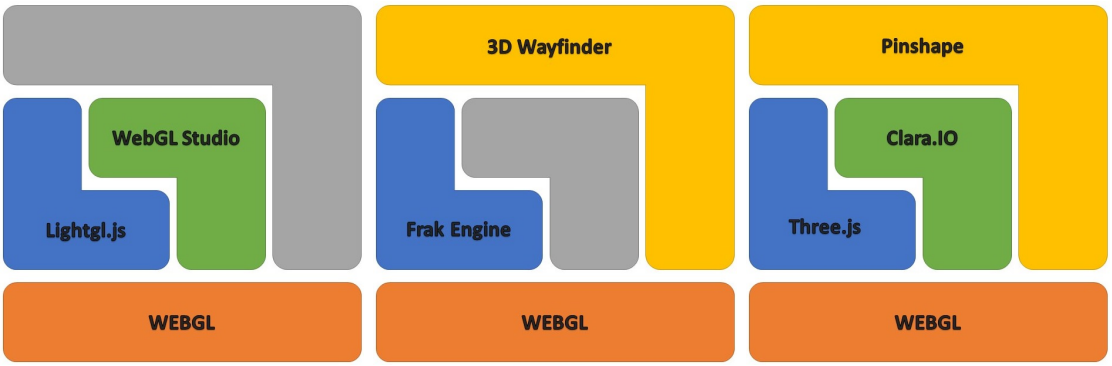

Figure 2.3: Graphic representation of different layering possibilities, adapted to three real case studies. The diagrams shows the interaction between (from left to right): basic library (Lightgl.js) and middle level solution (WebGL Studio); basic library (Frak Engine) and high level application (3D Wayfinder); basic library (Three.js), middle level solution (Clara.io) and high level application (Pinshape).

all solutions, is also related to the will to promote a transversal analysis over the possibilities offered by the main approaches. In this way instead of producing a linear list of libraries/tools/apps, we aim at obtaining a more useful comparison of different philosophies and methodologies. Moreover, we also think that this decision may be the most suitable in providing with a full understanding of the breadth and depth of each contribution developed for publishing 3D content on the Web.

2.3.2 Characterizing and Grouping the Web3D Features

Characterizing the different approaches by their features means, first of all, to define what are the atomic actions and functionalities needed for publishing and interacting with 3D content online. To this aim, we have identified a large set of characterizing features that should be able to satisfactory cover the possibilities provided, supported or needed by a Web3D solution.

This set turned to be a long list of disparate aspects, including features at different levels, for both implementation and usage:

- supported data types and associated representation scheme (3D volume, particle systems, triangle-based, etc.);
- publishing modalities (developers aimed, hybrid node-based, for naïve users, etc.);
- scene creation and organization tools (hierarchical structures, procedural definition, geometry instancing, etc.);
- hyperlink-based integration between 3D and other media (text, images, etc.);

- data encoding and transmission schemes (LoD formats, progressive streaming, view-dependent refinement, etc.);
- provided object/scene interaction paradigms (inspection vs navigation);
- scene customization possibilities (shaders, materials, lighting maps, etc.);
- supported input/output modalities (touch-based, gesture-based, WebVR, etc.);
- data pre-processing possibilities (client-side vs server-side);
- informative scene enrichment tools (annotations, hot-spots, etc.);
- supported publication aims (pure visualization vs digital content creation);
- data IPR management modes (watermarking, grant permission, access passwords, etc.);
- scene-level interaction elements (toolbars, 2D maps, view-cube, etc.);
- annotation/hot-spot authoring tools (policies for implementation and use; pro e cons);
- scene animations support (camera animations vs model animations);
- supported publishing experience (specialized analytical tools, distintive interaction paradigms, community aimed features, etc.);
- data hosting costs (disk space footprint, payment fees, freemium plans, etc.);
- data storage and access protection (data encryption, data center secure location, threat prevention, etc.);
- distribution terms and costs (open-source, freemium, commercial, etc.).

Given the variety of features, instead of building a hard-to-read (and full of foot-notes) table, we decided to structure the discussion of the following sections by grouping the features in few sensible macro-groups, related to their scope and usage. This harmonization process defined the following five groups (also schematized in Figure 2.4):

DATA LEVEL = functionalities related to the 3D data handling, ranging from the *transferring schemes* to the *rendering modalities* adopted to efficiently port and represent the 3D models. This level also introduces higher-level *policies for data management*;

SCENE LEVEL = functionalities needed to define a structured scene (as a *composition* of a number of basic 3D components) and to personalize it (modifying for instance the *shading* techniques). This level includes the possibilities to manage the interaction between those components (like in the case of scene *animations*);

INTERACTION LEVEL = functionalities concerning the interaction of the final user with the represented 3D scene/object. This level introduces also the *interfaces* and the elements required to drive specific input devices and advanced output modalities (like in the *WebVR* case);

INTEGRATION LEVEL = functionalities implemented to provide a full integration between the 3D content and the other multimedia content present in a webpage. This level investigates all the linking and embedding capabilities, ranging from *annotation* tools to *hot-spot* definitions;

PUBLICATION LEVEL = publishing functionalities analyzed at the higher abstraction level. This includes the more general concepts of publishing target, intended as modality of publication provided (*node-based*, *coding-aimed*, etc.), type of publishing promoted (*assisted*, *collaborative*, etc.), or class of experience supported (directed to *analysis*, *community* oriented, etc.).

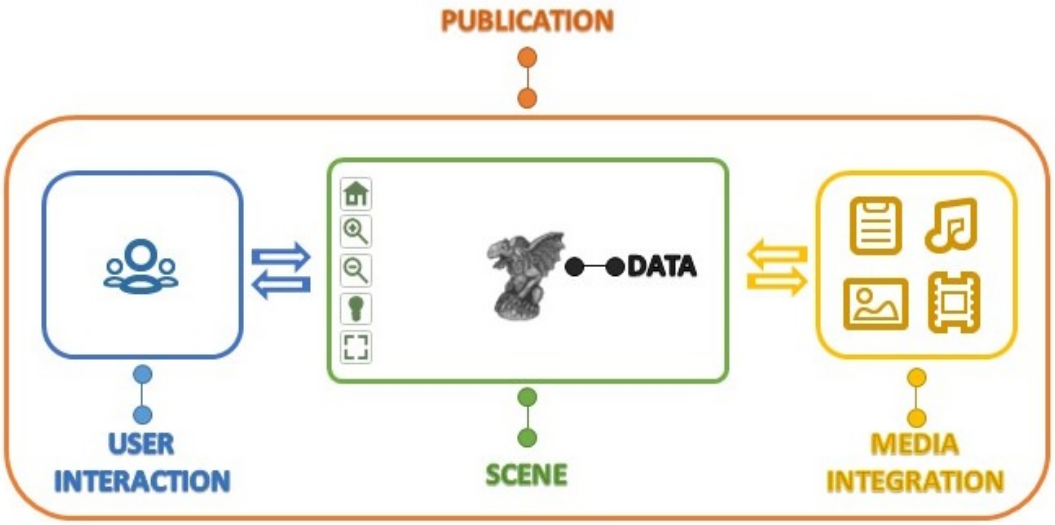

Figure 2.4: A graphic representation of the five macro-classes of features defined. The scheme proposed covers the whole ecosystem of Web3D solutions, ranging from low level to higher level functionalities.

Obviously, each of these groups cannot be considered an absolutely isolated container. As expected, also in this case the boundaries of these groups are fuzzy,

and many of the discussed features often exist only if strictly correlated with characteristics listed in a different group of features. At the same time, some other features could be spanning across more than one single group. But, in the end, this simple schema, considering five areas commonly shared by almost all the Web3D solutions we analyzed, proved to be an effective way to analyze the whole landscape of of the published 3D content online.

2.4 Analysis of the Features

In the previous sections we introduced a *per-feature* presentation and categorization approach. After listed a potential set of features, we established five macro-groups, to better organize the discussion. The following sections will thus be aimed at discussing each of these macro-groups, detailing each feature and providing relevant example of how the same capability/issue has been addressed by the different existing Web3D solution.

2.4.1 Data Handling

To discuss the Web3D publishing features we follow a logical order, starting from the innermost component of the Web3D tools and services, which is the *data* level. Analyzing this level we will touch both low-level features (data representation types, rendering techniques, transferring schemes, pre-processing optimizations, etc.) and higher-level functionalities (storage policies, IPR protection, etc.).

Studying the Web3D context, the first thing that catches the eye is the difficulty to manage inherently complex data in a computationally poor environment, such as the Web. Over the time, this issue has led to systems tailored to the specificity of the data to be managed, proving that the data involved in the publishing process, and the related design choices, play a central role in characterizing each Web3D solution.

While approaching Web3D publishing, content creators face a preliminary basic characterization, strongly dependent on ***data types*** and ***representation schemes***. Indeed, the visualization of different dataset would require different representation and rendering techniques, influencing also the choice of the proper publishing channel.

For instance, *volume visualization* (very popular in medical applications, but also in geographic and meteorological visualizations) often relies on *volume ray-marching* rendering algorithms, i.e. GPU-based techniques that use textures for data storage (MEDX3DOM [Con12] and X Toolkit [HRA+14, XTK12] are two examples of medical solutions adopting this rendering approach). If, conversely, the aim is to visualize a *particle system*, it would be preferable to use the *ray-casting* technique (another GPU-based technique, that defines objects using implicit primitives, and then generates their surface by computing ray-object

intersections). Examples that fruitfully exploit this approach can be found in the biomedicine/molecular visualization domain, where atoms are approximated with spheres and center/radius of these spheres are the parameters to send to GPU (examples include the systems by Mwalongo et al. [MKB+15] and Chandler et al. [COJ15]).

Even though data-specific rendering techniques are frequent, recent advances in Web technologies (once again, JavaScript improvements and WebGL availability) have made the *triangle-based* approach more and more efficient, making it the most widely used approach, not only when "classical" mesh-based 3D models are involved, but also extending this representation to the visualization of other types of data: geospatial maps [JvL16], city models [GVB+15], marine data [RWW14], and many others, including the very same volumetric 3D datasets [HKBR+14] and the particle systems [RH15] previously mentioned.

However, despite the increase in performances due to the adoption of GPU-enabled rendering solutions, the efficiency of the rendering system remains tightly coupled with the intrinsic characteristics and granularity of the data, thus maintaining relevant the importance of a keen choice of the data representation and rendering strategy.

Strongly related to the adopted rendering techniques, ***data encoding*** and ***transmission approaches*** induce another basic characterization of the Web3D systems. As already stated in §2.2.2, an efficient data handling is fundamental for interactive Web visualization, mainly because, without a performing data transfer, low bandwidth and latency issues can lead to long waiting times until data is available to the browser.

As we have discussed, a number of algorithms have been formalized specifically to face this bottleneck, mostly proposing *transfer formats* able to support a *progressive streaming* of simplified versions of the handled 3D geometry, such us P3DW (derived from [LCD13, LCD14]), or SRC (derived from [LTBF14] and used in [Fra09a]).

Some of those representation schemes also provide additional features for LoD or multi-resolution encodings for progressive rendering. They may include network-oriented compression and decompression algorithms, that ensure efficient decoding and rendering rather than only optimizing the compression ratio, like it happens in the WebGL-loader [Goo11c, Chu12], a solution developed in the context of the Google Body project [BCK+11] (based on UTF-8 coding, delta prediction and GZIP, produces compression ratio of around 5 bytes/triangle for encoding coordinates, connectivity and normals). Other optimizations related to progressive rendering techniques concerns the refinement criteria: they may be view dependent (refine the geometry resolution depending on dynamic camera specifications) or based on the system rendering load (provide a high resolution representation for more static visualizations, and a lower resolution for more demanding computational conditions, as in the example presented in Figure 2.5).

Nowadays, almost all the main Web 3D viewers adopt LoD/multi-resolution representation schemes (often developing proprietary transfer formats).

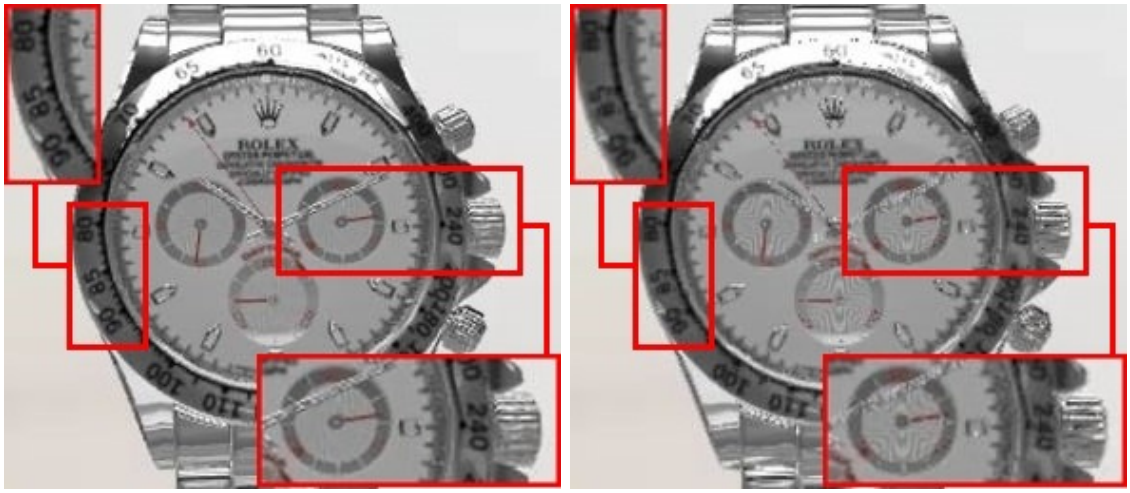

Figure 2.5: Example of selective LoD rendering driven by animations or user interactions in the ThreeKit [Hou15] viewer (an online 3D configurator based on discrete layered models, built on the top of the Clara.io viewer introduced in §2.3.1). Whenever the final user starts to interact with the model the system switches from a higher (left) to a lower (right) resolution representation, to keep pace with the more frequent image refresh rate required by the interactive session.

Nevertheless, for specific cases where the original coherence/precision of the 3D data is more important than minimizing the download times, or when the kind of primitives are difficult to simplify, single-resolution representation are still adopted. An example is Autodesk Tinkercad [Aut11] (a browser-based 3D design and modeling tool oriented to 3D printing), as well as most of the Computer-Aided Design (CAD) solutions. Consequently, those system expose the final user to considerable data transmission times in spite of a solid precision of the downloaded 3D data.

Another justification for using a single-resolution approach is that the conversion of a model to a LoD/multi-resolution scheme is usually a bit complex in time, and thus improper to be done on the fly in the case in which we are dynamically changing the base mesh (such as in a modeling session of a CAD tool).

Indeed, the main drawback with LoD/multi-resolution approaches is that they require heavy 3D ***data pre-processing*** before the Web publishing. Data pre-processing, however, is often necessary for highly specialized viewers (like the ones mentioned at the beginning of the section), which, relying on highly optimized and custom-tailored rendering, generally require converting data to specific, optimized formats. This step can be performed locally on the client device or, rather, remotely (usually performed by the server that hosts the Web3D publishing service). Both approaches have pros and cons.

In *client-side* conversion the data optimization is performed on the client, thus

the content creator needs to download and run additional software for converting the 3D data into a layered or multi-resolution format. Converting the data on the data-owner device prevents to upload online the often huge single-resolution 3D model (usually, the progressive encoding is compressed and, thus, the final encoding is much smaller than the original data file).

In the *server-side* case instead the data conversion process is run on the server: it is just required to upload the data file and no clues on the data conversion process are provided. The pro is that content creators should not be aware of the technicalities, should not bother ensuring availability of the required resources (memory space or processing power) and do not have to install the required software tools for data conversion. Therefore, this approach increases the perceived degree of automation of the Web publishing process.

For those motivations this latter option is typically preferred to the first one by end-products supporting the online 3D publishing in the form of one-click publishing services. An representative example of solution adopting this approach is the Sketchfab platform [Ske14] for community sharing of 3D model, that converts data to a rendering-optimized and compressed representation. However, especially if uncontrolled and undocumented (like often it is in commercial systems), data conversion can somehow affects data integrity. This may be perceived as a quality degradation or even represent an issue by some specialized applications field (such as medical diagnostics data visualization and analysis).

The migration of 3D data towards Web friendly representation formats (simplified and/or compressed) is fundamental when storage and data transmission requirements are critical issues (as it is the case of service-oriented platforms). So, also considering the peculiarities of 3D data, the ***cost of models online*** (intended as disk space footprint) can easily become a relevant discriminant in the choice of the most appropriate publishing approach. Usually, commercial Web3D platforms that offer publishing space to share 3D creations, propose to their customers freemium or payment plans diversified on the required storage space or the provided rendering quality (e.g. PlayCanvas [EEKI+11], a game engine for creating, publishing, and sharing on the Web interactive 3D applications). Disk-space and bandwidth efficiencies remain a central issue also when working with open source solutions (which usually let the developer embed in its webpage the 3D viewer for free), since also in this case the server storage and bandwidth resources are usually outsourced to an external fee-based service.

The policies adopted for 3D data encoding have a potential impact over another important Web3D feature, the ***IPR management*** issue. Indeed, exploiting LoD/multi-resolution and view-dependent techniques may represent not only an advantage in term of performances, but also a sort of implicit data protection. Since with these approaches the full 3D model is never sent for rendering in its entirety (in fact Web3D systems send to the client only model chunks customized over the current view specifications), the fraudulent copy of original data would need more advanced methods (reverse engineering techniques).

But when LoD/multi-resolution is not needed and the 3D object used has a "plain" geometry, then IPR protection becomes a problem. This crucial point, common to all the media-hosting services, in recent years has also become relevant in the Web3D world, driven by the explosion of sharing platform for 3D printing (Thingiverse [SP11], Shapeways [Sha13], Threeding [Thr13], MyMiniFactory [MyM13], etc.). IPR protection of online 3D data is a hot topic for Web3D, 3D watermarking can be a (partial?) solution, but anyway the current approaches to build a defense are often based on set of progressive barriers. Many solutions try to protect the uploaded content from unauthorized replication (by other users), for example giving the opportunity to make published material private (non indexed, or accessible only through temporary links) or not-downloadable. In some cases the adoption of proprietary file formats (preferably LoD/multi-resolution, as we have seen), or the use of password-protection for every model, can add additional security to the management of the uploaded 3D content. The ShareMy3D [AH15] platform for publishing (and storing) 3D meshes online is an example of system offering all these features (although only for paying users).

The characterization of the security policies of the Web3D approaches should also take in account the more general and higher level ***data storage and access protection*** (i.e. related to the cloud servers used). Nowadays these infrastructure should be compliant with specific ISO Standards for Information Management Security (ISO/IEC 27001), able to ensure a number of security mechanisms, including the physic *data centers secure locations* (perimeter fencing, patrolled security guards, biometric entry authentication, laser beam intrusion detection, etc.), *data encryption* (automatic encryption under a 256-bit Advanced Encryption Standard, regularly rotated set of master keys, etc.), and the *threat prevention* (uninterruptible power and backup systems, fire/flood detection and prevention, etc.).

2.4.2 Scene Setup

In the previous section we analyzed the core features of the 3D data management. But that what is presented to the user is often not just a single 3D model, but a more complex set of entities. Consequently, most 3D viewers rely on the concept of *3D scene*. A scene is, basically, a "container" defined in three-dimensional space, used by the 3D application to arrange all the entities that are needed to represent, manage, display and interact with a three-dimensional environment. To this extent, it may be considered the atomic unit of a 3D publishing project. A scene contains one or more 3D entities (mesh models, but also particle systems, impostors, volumetric data), arranged in a reference space, plus all their related assets (textures, materials, shaders, animations), but also contains all the entities used to compute their appearance (such as lights), one or more cameras (the point of view of the user), and all the other 2D and non-dimensional entities used for

interaction and rendering.

This section presents the *3D scene setup* possibilities provided by the various Web3D approaches, including scene appearance (color, reflectivity, transparency, etc.), logical and spatial organizational structures, composition customizations, etc.

Many applications require the 3D scene to be organized into some kind of ***hierarchal structure***, which is used to organize the different elements of the scene following a spatial or logical arrangement.

Scene graphs are the most popular among this kind of structures: they organize the scene as a complex graph that includes all the elements of the scene as a set of nodes, possibly resembling an identifiable rooted tree. Scene graphs approaches are frequently used in 3D rendering engines, and are generally based on a logical, rather than spatial arrangement of the data nodes. Many scene graphs also provide *geometry instancing*: a system in which the scene graph nodes represent entities or objects in the scene, that refer to a single copy of the data (made up of a 3D mesh, textures, materials, etc.) kept in memory just once. This allows to reduce memory budget and to increase rendering speed, but also to organize and manage the handling of physic interactions, collisions detection and large scale animations. For these reasons, scene graphs are particularly useful to manage increasingly complex 3D scenes, for instance, those of modern 3D games.

Although there exists specific libraries (e.g. OSGJS [Pin11]) developed to port scene graph concept in WebGL, nowadays many Web3D approaches are based on proprietary inbuilt scene graph managers. This tendency follows the same idea expressed in the previous section, of tailoring the data management to the specific characteristics of the target data and the specific application, sacrificing generality for more efficiency. Examples of solutions adopting this approach are: SceneJS [Kay10], an open-source low-level scene graph based engine for 3D visualization; or, at a higher level, the already mentioned Unity3D [Uni05] engine (a freemium multi-platform authoring tool, initially designed for 3D games but currently used for generic purpose interactive 3D creations and installations).

However, scene graphs are not the only strategy followed in the Web3D environment and, in some specific situations, these logical-hierarchal structures may also complicate the content creator life, for instance, enforcing a higher level of hierarchy, when a much finer, or spatially-aware level of control over the execution flow is instead needed. Therefore, some solutions, such as the already introduced Lightgl.js [Wal11] or Stackgl [Sta15] (a WebGL software ecosystem inspired to the Unix philosophy), do not explicitly provide a scene graph, and leave to the content creator the freedom to built it on top of a basic functionalities layer (just like the previously mentioned WebGLStudio [Age13] system does, featuring scene graphs despite being based on Lightgl.js).

A frequent alternative to the support of the scene graph concept is to enable the ***procedural definition*** of the 3D scene. This approach exposes several fundamental components, then combined in a procedural way to form more complex entities. Solutions adopting this composition scheme (which also includes

a geometry instancing system), propose a more flexible way for creating content-rich scenes and representing 3D objects, offering a performance-wise data structure that can be effectively used in applied research or algorithm prototyping. An excellent example of this kind of approach are those systems in which the goal is not to present a large 3D scene, but rather to find a way to visualize efficiently very complex 3D datasets, such as the Potree system [Sch13], an open-source viewer for large point clouds, that uses multi-resolution octree-based algorithms for displaying at interactive rates enormous unprocessed 3D point clouds.

A structured 3D scene can also include the definition of the ***visual representation appearance*** of its 3D content and, in some cases, it may also allow for a complete run-time control of the rendering appearance at object or scene level. It may be argued that, in some contexts, the appearance (material, texture, shader) should be considered part of the 3D model data. However, more often than not, in Web3D applications the shaders, materials and textures are assets that are associated to the 3D model only at the level of the scene, and mostly at publishing time; moreover, the possibility of changing in real-time the rendering appearance of the scene elements is certainly a scene-level functionality. This "soft" link between 3D data and appearance certainly derives from the nature of the WebGL rendering pipeline, that is completely based on shaders.

To customize the 3D scene visual representation appearance involves the ability of a Web3D solution to modify the GPU rendering pipeline, personalizing the adopted *shaders*, *materials* (mapping enhancements that allows for objects to simulate different types of realistic materials), and *lighting maps* (techniques used to create various rendering effects defining different types of light sources).

For a Web publishing software, the more this rendering stage is programmable/configurable, the more it is easier to achieve complex visual effects, at the cost of requiring a much deeper knowledge of rendering concepts from the the content creator (shifting the bar from Web programming to CG programming expertise).

Some Web3D solutions provide a complete set of customizable parameters, comparable in all and for all to the classics stand-alone 3D creation suites. BabylonJS [CR13], a relatively new open source JavaScript/TypeScript 3D engine oriented to game design, is a good example of those. It allows developers to fully personalize the rendering/shading process by creating custom shaders, materials, and lighting; providing explicit features to set: diffuse, ambient and specular lightning; opacity, reflection, mirror, emissive, specular, bump, and lightmap textures; unlimited lights (points, directionals, spots, hemispherics); Fresnel term for diffuse, opacity, emissive and reflection; etc.

In this kind of solutions, usually the set up of the 3D scene appearance is defined simultaneously with the geometrical scene definition. Some other Web3D applications also provide the possibility to change the initial settings in real-time during the visualization, using external variables or textures for modifying the algorithms defined in the shaders. Obviously, in this case the set of actions allowed

to the final user is considerably smaller than in the previous case.

The already mentioned Sketchfab [Ske14] publishing platform, offers the possibility to choose between few predefined shading methods, each one with a series of customizable parameters. The content creator, at upload time, beside configuring the optimal shading method for its 3D model, may also choose which method and parameters will be available to the content consumers, when exploring the scene/object.

Conversely, the Smithsonian X3D [Smi11] viewer (developed by Autodesk for this cultural institution), has a single shading pipeline but offers to the final user a lot of real-time editable shading parameters, material properties, and the control over multiple lights position and color.

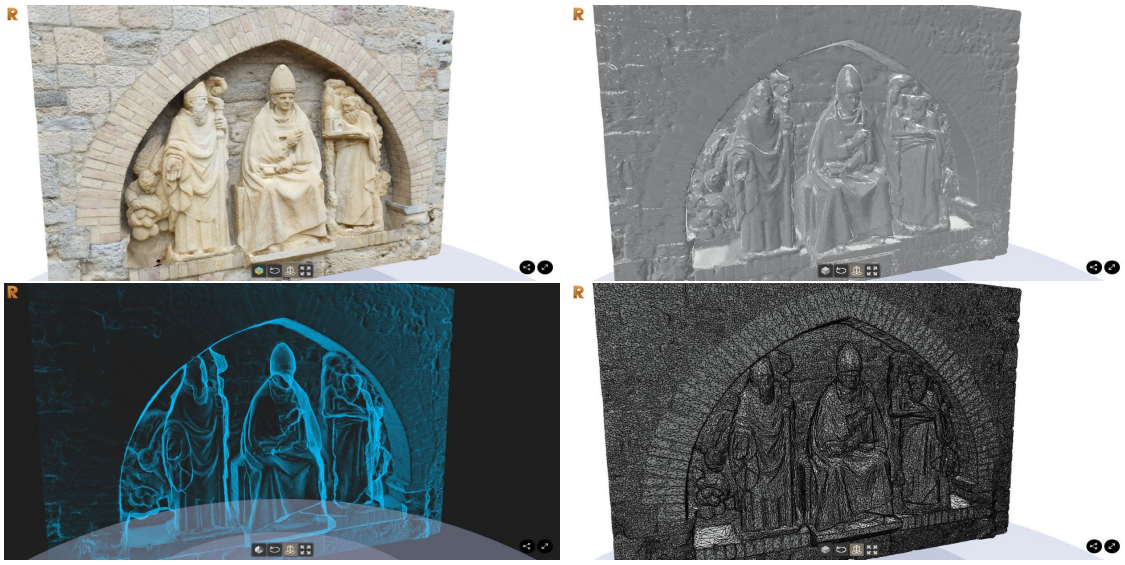

Figure 2.6: Four rendering possibilities provided by Autodesk ReMake [Aut15] (in clockwise order starting from the upper left): textured, solid, wireframe, and X-Ray mode. Interacting with the viewer controls the final user can select in real time to visualize one of these options.

ReMake [Aut15] (again by Autodesk), an end-to-end solution for creating (offline) and publishing (online) 3D scenes, provides a single shading method, and only offers the chance to switch on the fly between a predefined set of rendering modes (see Figure 2.6).

As said above, a 3D scene also contains the specifications of a *viewpoint* or *camera*. This element is not only important for rendering the scene, but also to specify others characterizing features, such as ***camera animations***. Camera animations may be: simple predefined views; smooth view interpolation between predefined positions; or pre-computed camera paths. Even if this kind of animations seem to be relatively simple (when compared with complex or large

scale animations of the 3D scene objects), they result extremely useful to transform a simple 3D viewer in an effective publishing tool. Thanks to their simplicity and high effectiveness, they are available in many systems. The availability of this feature allows the creation of bookmarked views or Points Of Interest (POI) inside the 3D scene. These predefined views can be linked to other Web media/components outside the canvas element, or also interconnected in a smooth, animated, and immersive view path inside the 3D scene: the Archilogic [Arc14] platform (focused on 3D architecture and interior design) provides a functional implementation of this kind of guided presentation.

Of course, cameras are not the only animation that is available as an asset in a 3D scene, and the more complete platforms (particularly the ones oriented to gaming) provide to the users features to animate all the elements in the scene. An exhaustive review of animations possibilities on the Web is presented in the recent work by Ahire et al. [AEB15]. However, for the sake of conciseness, the main approaches to 3D online publishing (such as the Sketchfab [Ske14] platform) support just these animation modalities:

- *skeleton-based*: the skinned 3D surface of the model is connected to an animated multi-joint structure, generally used for character animations;
- *rigid*: animate model translation, rotation and scale, generally used for mechanical animations;
- *morph to target*: morph shapes from one state to another, generally used for facial animation.

The availability of camera and objects animations is a fundamental requirement while evaluating different Web3D solutions, because by means of these features the content creator can cross the borders of the 3D rendering container, introducing elements of *digital storytelling* and moving the published content towards an interactive and integrated experience. The "Experience Curiosity" [NAS15] (a 3D online serious game shown in Figure 2.7) is a brilliant and simple storytelling example achieved combining together camera and 3D model animations; developed by NASA's Jet Propulsion Laboratory using the Blend4Web [Tri14] publishing solution; it was presented at the WebGL session at SIGGRAPH 2015 [Khr15b].

2.4.3 User Interaction

Strongly connected to the scene, the *interaction level* is one of the most characteristic features of a Web3D solution. While the features at this level may be tailored to specific types of 3D data/content/application, we can envision, also in this case, some common elements and some characterizing features. User interaction in a Web3D solution happens at different levels: it includes features supporting interaction paradigms that works at object or scene level (trackballs,

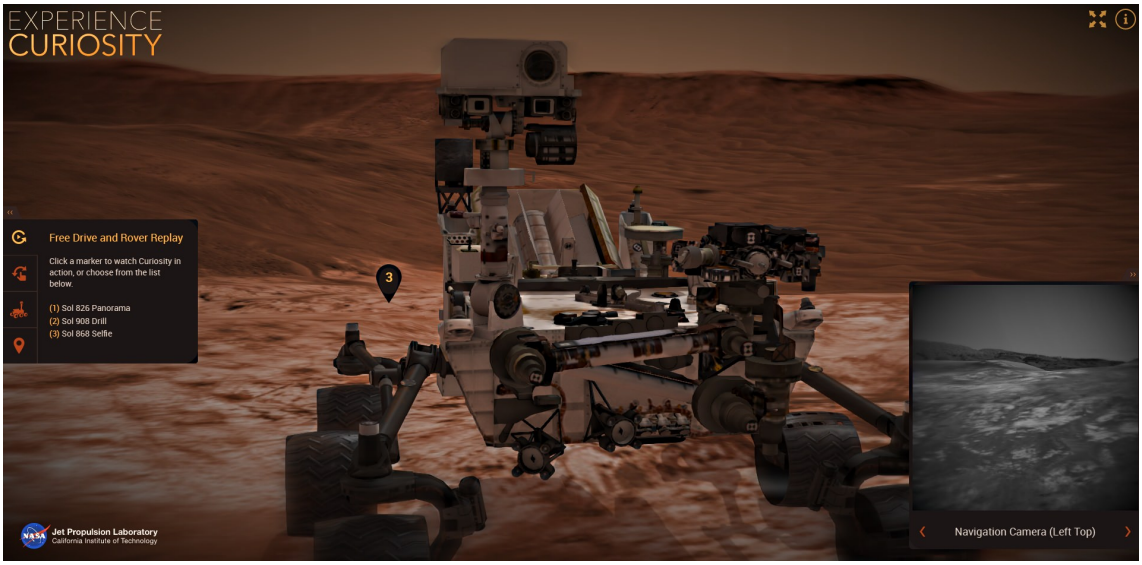

Figure 2.7: The interactive 3D Web application developed by NASA exclusively using open-source software (including the commercially licensed Blend4Web [Tri14]). This application was released in 2015 to celebrate the 3rd anniversary of Curiosity rover landing on Mars. The final user can control both the rover and the camera, driving the rover across the bookmarked views placed on the virtual scene (marker number 3 in Figure).

first person controllers, manipulators, etc.), but it is also present as interface elements focused to the interaction between the webpage and the 3D layer (toolbars, hypertext-based functions, etc.), or it even may concern the interaction of the Web3D environment with novel technologies and input/output devices (touch surfaces, VR devices, gyro sensors, etc.).

Interaction with 3D Content

In the last two decades, several research groups have studied an effective interaction with 3D objects or scenes [Han97, BKLP01, BKLP04, JH13, JH15], leading to design approaches able to cover most of the relevant combinations of data and workfields. Nowadays many of these interaction approaches have been transposed online, implemented in the various available Web3D solutions.

The interaction in a 3D environment can be characterized in terms of *universal interaction tasks*: the process of getting around a virtual environment is one of these. In the variety of existing interaction paradigms deriving from this task, *exploration* approaches addressed to general movements are the most exploited. In particular, two main exploration techniques are widely supported in 3D Web systems:

- the interactive *inspection* (rotate/pan/zoom), where the user interaction moves the object/scene in front of the camera;

- the interactive *navigation* (walking/driving/flying), where the user moves the camera through the scene.

This dichotomy is often called *World In Hand (WIH)* vs *Camera In Hand (CIH)* [WO90, GCPDB14].

Interactive inspection. Rotating, panning and zooming are the basic view movements, used in almost every 3D software. These operations, applicable at the object level and at the scene level are also extremely common in the Web3D world because, being easy to map them to two-dimensional operations, they work well with common pointing devices (such as a mouse). This paradigm generally implements the concept of a *virtual sphere* containing the object to be manipulated, also known as *trackball*, a seminal technique designed for moving around 3D objects (based on the Chen et al. [CMS88] Virtual Sphere and Shoemake [Sho92] ArcBall).

This interaction method is perfect for solutions aimed at the inspection of a single object, or of simple scenes (like in the case of the viewer developed by the University College London for the 3D Petrie Museum [Uni09] project), where the object/scene can be fully explored with the camera from the outside, looking towards a center of interest.

On top of this basic behavior, it is possible to implement more sophisticated and customized interactive inspection modes, maybe directly related to the characteristics of the object to be inspected, or to the application field. The 3D Web viewer developed, for instance, in the Visionary Cross Project [Vis15a, LCD+15] (an international cross-disciplinary project aimed to exploit recent developments in the Digital Humanities to study of a key group of Anglo-Saxon texts and monuments), adopts a specific trackball that allows to inspect the elongated object present on the virtual scene only through constrained panning movements.

However, despite most systems do provide an interpretation of the virtual sphere/trackball, a standardization of its components and behavior is still missing. In fact, moving from one 3D enabled application to another, usually the behavior of the virtual sphere doesn't remain concordant, and small details (such as keys/buttons mapping or inertia) are often different.

Interactive navigation. On the other hand, when the scene or the 3D models represent a 3D environment, it may be useful to let the camera viewpoint travel inside the scene, to bring its point of view in relevant and natural positions. Therefore, the other common approach for allowing final users to navigate a 3D environment is the *walking/driving/flying paradigm.*

One interpretation of this approach, directly derived from the video game development world, corresponds to the *First Person Perspective* (FPP) navigation, where the view position of the player is usually controlled with arrow keys (for this

reason, also known as WASD paradigm) and the view direction is controlled with the mouse. Implemented by all the game-oriented Web3D solutions (like the previously introduced PlayCanvas [EEKI+11]), despite being versatile and powerful, it requires multiple simultaneous inputs, and it is often deemed too complex for some classes of final users.

Considering that these interaction paradigms are highly specialized on the handled dataset and on the experience that the publication wants to convey, since the Web3D beginning multiple options have been provided by the various solutions. As an example, VRML [Rag94] already provided the possibility to set the navigation node (the element containing informations describing the physical characteristics of the viewer's avatar and viewing model) with at least four different types of user interactions: "WALK" (interactive navigation with gravity), "EXAMINE" (interactive inspection), "FLY" (interactive navigation without gravity), and "NONE" (interactions disabled, system in guided tour mode).

Interactive inspection and navigation modalities are widely distributed because of their effectiveness for general cases, but many other more specialized paradigms are available on the Web. For example, 3D maps softwares, like the popular Google Earth [Goo11a] application, usually provide *POI logarithmic movement* (a targeted interaction for smooth shifting, also known as Go-To/Fly-To), while CAD systems and 3D editing softwares (like WebGLStudio [Age13]) generally use specified coordinates (x, y, z points supplied by the user via dialog boxes, or other kinds of 2D interface) both for positions and orientations displacement.

Specific coordinate movements are also the base for the *selection and manipulation* task, interaction technique different from the approaches presented so far, consisting in choosing an object and specifying its position, orientation, and scale through explicit and direct translation, rotation, and scaling tools, called *manipulators* [Bie87, NO87, CSH+92, SC92]. Especially used during scene construction in 3D modeling softwares (like Autodesk Maya [Aut98]) and game development environments (like Unity3D [Uni05]), this paradigm results very efficient in designing 3D scene with multiple objects, in which users have to repeatedly realign and adjust different parts. Nowadays the implementation of this manipulation technique almost always relies on visible graphic representations of the operations on (or the state of) an object displayed with the 3D model and able to control it via graphic interactions [SIS02].

The interaction with 3D content may also happen on another level, i.e. characterizing the system through the *application/system control* task, which describe the interactions in a virtual environment in term of communication between a user and a system, which is not part of the virtual environment. This technique, widely exploited in 2D "point-and-click" *WIMP (Windows, Icons, Menus, Pointer)* graphical user interfaces, has been adapted also in 3D applications, where control interface components are placed in a conventional 2D interface presented in screen space on a 2D plane called *HUD (Head-Up Display)*.

This WIMP application control approach splits the interface into two parts with very different interaction metaphors, since the navigation and manipulation functionality is accessible through the 3D scene while the rest of the environment controls can only be interacted through the screen space interface overlaying, or side-by-side with, the 3D scene.

The online game World of Warcraft [Bli04] represents a successful example of this interaction approach, interesting also because the implementation follows the design concept described as "layered approach to learning" [Shn83], in which, to overcome the usability issue and optimize the learning curve, at the beginning of the game the interface is quite simple, but it acquires more functionality and elements as the players gain experience.

Indeed, if on one hand the availability of all these possibilities is a resource, on the other hand it also makes steeper the learning curve for more naïve users, which often find it difficult to manipulate and interact with 3D environments. Quite often, this is the result of a non properly designed user interface, unable to efficiently associate these paradigms and the represented scene, or unable to conform the interaction to what is expected in the publishing target ecosystem.

Web-Based Interfaces

In an ideal world, the selection of the best interaction mode should be tightly connected with the handled 3D content and with the publishing purpose. But in a Web3D system, it should be also tailored with the Web characteristics, possibly adopting techniques that are optimal for the *hypertext-based* Web interface.

From a research point of view and focusing on the more recent years, Jankowski probably has been the most active in trying to bring together hypertext technology and interactive 3D graphics. He aimed at clarifying some of the foundations of 3D Web user interface design [Jan11], focusing on an understanding of the fundamental tasks users may be engaged in while interacting with Web-based 3D virtual environments, and then introducing his interface management approach. His work proposes a Dual-Mode User Interface [JD12a, JD12b] that follows the usual hypertext-based interactions mode, with the 3D scene embedded in the hypertext, but also exploits an immersive 3D mode, which moves the hypertextual references directly into the 3D scene.

Indeed, although hypertext-based Web interfaces are optimal in linking together Web and 3D (implementing a step towards the essential integration between 3D and other media, as we will see in §2.4.4), in some situation they may not be enough in providing a full interaction with the 3D scene. In these cases toolbars and other clickable graphical commands/elements/buttons immersed in the virtual scene (or, as we have already seen, superimposed in screen space), are a practical solution. This is nowadays provided by a large number of systems: Autodesk Tinkercad [Aut11], Clara.io [Exo13], Archilogic [Arc14], etc. They usually make available a set of actions aimed to help interactions with the 3D

scene, orienting the final user in using the adopted exploration paradigm (without preventing to jointly exploit the hypertextual approach).

Two representative examples of this class of additional elements enabling an improved interaction are: the *overview-plus-detail* components (allow to simultaneous display an overview and a detailed view of an informative space); and the *orientation widgets* (help to address the problem of disorientation, especially in WIH interfaces). Both of them are supported and exploited in a good number of Web3D solutions.

2D MiniMaps are a good example of the first group of overview-plus-detail elements. Often superimposed in a corner of 3D viewers addressed to interactive visualization of planar dataset (terrains, floor plans, geographical systems, etc.), these alternative media support easier transitions between different locations of interests, providing an improved self-localization of the final user in the represented space (see the Potree [Sch13] example in Figure 2.8).

Figure 2.8: The Potree [Sch13] basic viewer showing a running example of 2D MiniMaps (in the top left corner of the image). This additional interface element is useful to georeference the 3D model, but also to provide a visual feedback of the camera (represented by the triangle shape in the map) position and orientation into the virtual space. MiniMap elements refresh their position every time final user interacts with the 3D scene.

A proper representative for the orientation widgets could instead be the *3D ViewCube* element. Introduced by Khan et al. [KMF+08], this cube-shaped component is both an orientation indicator and controller generally placed in a corner of the scene window; the Smithsonian X3D [Smi11] is an example of 3D Web viewer using this interface component (see Figure 2.9).

Additional interface elements like these, can be also exploited by Web3D solutions to develop and provide specialized features aimed to specific application

Figure 2.9: Smithsonian X3D [Smi11] is an example of 3D Web viewer using the ViewCube interface component (in this case presented at the top right corner of the 3D rendering window). The 3D cube widget rotates synchronously with the 3D model; it has the primary purpose to provide the final user with a feedback on the orientation of the 3D scene. The component can be used also to directly drive the scene interactions (by rotating the 3D cube the final user applies a corresponding rotation to the 3D scene).

domains, some of which often requires more technical or analytical presentation tools. A system designed to present Cultural Heritage (CH) 3D artifacts, for instance, usually need features able to support annotations and metadata (like the Aton [Vir15] front-end software has), or even technical measurements and visualization of sections, like the ones provided by the aforementioned Smithsonian X3D [Smi11] platform. A solution addressed to handle 3D chemical structures instead, possibly should provide tools for visualize vibrations, orbitals, schematic shapes, and symmetry operations, as well features for assisting the management of molecules, crystals, and materials (like JSmol [Jmo13], a browser-based biomedical viewer, does).

A general issue with these screen space controllers is often related to the user immersion in the 3D context [Han97]. Indeed, requiring to switch from interacting directly with 3D objects to indirectly interacting with them, these interfaces may easily lead to lose the user engagement (a relevant problem in more immersive 3D applications). To overcome the "cognitive distance" due to map 3D tasks and 2D control widgets in a 3D space, in 1997 Van Dam [vD97] introduces new user interfaces (not dependent on classical 2D widgets such as menus and icons) called "post-WIMP", which in the last years have reached also the Web3D [BCR05] world. These interaction techniques strongly rely on the latest I/O devices and technologies, like for instance, gesture and speech recognition, eye/head/body tracking, etc.

Specialized I/O Devices

For governing the interaction, modern 3D Web environments exploits common general-purpose hardware devices, like the mouse and the keyboard, but also more recent *input/output technologies*, like touch or multi-touch input surfaces. The extremely rapid market penetration of the latter kind of devices, occurred mostly because of the widespread diffusion of mobile devices, is redefining the way people interact with digital content. Now, thanks to the last generation of mobile devices, equipped with browsers supporting WebGL, this technological evolution is beginning to involve also the 3D medium.

As most of the interaction methods in the Web3D environment derive from (older) PC interfaces, they are undergoing a radical redesign process (like in [Mal13]) in order to enable also touch-based input devices. However, if it is true that these novel technologies open new possibilities, they also lead to new challenges and issues.

For instance, touch inputs favor direct and fast interaction with the manipulated content, but at the same time introduce some constraints: with respect to a mouse and keyboard, they possess a much smaller expressiveness (even considering gestures, we are far from the multiple keys and modifiers) and a lower precision (a mouse positioning can be pixel-perfect, a finger touch cannot). Moreover, they also require creating "switching" versions of the source code to enable multiple input/output devices configurations, that is an open issue particularly relevant for systems requiring to operate on very different hardware platforms, such as Web3D software does (the Seo et al. [SYCK16] work with super multi-view autostereoscopic displays is just a confirmation of that).

Nevertheless, touch based systems represent just the tip of the iceberg of new I/O technologies, especially considering the increasing number of novel devices, sometime specifically addressed to 3D applications, that are making practically mandatory to redefine the obsolete WIMP paradigm. Among these, there are the new devices supporting *gesture-based interfaces* by the means of fingers, hands, or body motions, which often do not require touch contact for generating an input signals, or that ones for enabling *immersive virtual and augmented reality*, or again the ones providing access to the wide set of sensors (gyros, accelerometers, etc.) available in portable devices.

All these new I/O methods, after being successfully tested in research applications (like [WWV+10], which presents techniques for exploiting the motion sensing capabilities of a console controller, the Wiimote by Nintendo, to enable a 3D user interface), and stand-alone devices (like Gravity Sketch [Gra14], a 3D creation tool focused on mobile and VR platforms that offers an innovative and intuitive design experience), are now arriving on the Web with device-mapping JavaScript libraries. A brilliant example of that can be find in the paper by Kwan [Kwa15], which describes how to use the *Leap Motion* device [BH10] (a low-cost commercial hardware device supporting hand/finger tracking) inside of the Three.js [Cab10] ecosystem.

Among these libraries, a good number are still working draft specification for the Web. It is the case of the *DeviceOrientation* [Moz16b] and *DeviceMotion* [Moz16a] event handlers for accessing orientation and motion sensors directly within the browser, or of the *WebVR* API [Moz16c] for providing support in generating *stereo pairs images* (couple of offset images that combined in the brain give the perception of 3D depth) for virtual reality devices, like the *Oculus Rift* [Ocu16] and *HTC Vive* [HTC16] head-mounted displays (HMD).

However, despite that, the chance to exploit new devices and I/O methodologies has been caught by several platforms, and, it is not unusual today to have access to Web3D implementations able to support them. Some examples are: Patches [Viz14], an online programming node-based editor expressly designed for building WebVR experiences; or Parallax [Tos12], a cross-platform Java 3D SDK that provides in its demo the possibility to switch between several different 3D effects: *anaglyph* (encodes each eye's image with filters of different colors to percept three-dimensional scene by composition), *parallax barrier* (consists of a series of spaced slits allowing each eye to see a different set of pixels to create a sense of depth through parallax), and the already mentioned *stereo pairs*.

2.4.4 Multimedia Integration

Over the years many initiatives approached the integration of 3D data in the standard Web publishing ecosystem. Several of them failed to reach their goal, mostly because they were focused on managing 3D content alone, keeping it isolated from other multimedia layers. Nevertheless, nowadays people working on 3D data online are aware of the need of a complete integration of the 3D content in the Web environment. Consequently, they also are aware of the importance to establish bidirectional channels able to logically and functionally link the 3D layer with the other media typically composing a webpage. Following this idea, this section will describe all the features aimed at transforming a simple embedding of 3D data online in a real integration of 3D content on the Web, discussing and comparing the strategies adopted by some pioneering Web3D solutions to breach the HTML CANVAS borders (like for instance hot spots deployment, annotation possibilities, etc.).

Designing a webpage it is possible to simply spatially arrange the various media inside it, but this would be a very limited and trivial take on the "multimedia" approach. While, for more classic media like text and images, their arrangement is fluid and shaped by the analogy with the composition/layout of a physical page, less static media like videos and 3D often are enclosed in rigid boundaries (or on dedicated single-page viewers). In particular, the integration of 3D is problematic, because the view area represents a window over a 3D space, while the rest of the webpage has a 2D nature.

A way to effectively connect the different media on the page is to try to exploit hypertextual links for connecting them, thus achieving a *trans-media storytelling*

behavior. Hyperlinks can be considered the game changing component of the Web, able not only to transform a linear and passive fruition of informations in an interactive navigation, but also to convert from spatial to logical the relationship between the elements on the page.

To enable this vision, a Web3D viewer should define a set of elements that may be useful to connect its components, events, and states with the rest of the webpage, for instance providing:

- A 3D analogous of a hypertextual link (e.g. a clickable geometry, a shape, or a marker) visible/highlighted in the 3D scene, that should be easily visually identified by the user.

- A series of exposed functions able to change the state of the viewer (e.g. the *change_model_visibility()* method to modify the visibility of an object on the 3D scene). These functions may be called by other components of the webpage to drive view/camera controls (maybe using the camera animations introduced in §2.4.2), data behavior, visual appearance, etc.

- A series of exposed functions able to read the current state of the viewer, or access some of its data (e.g. the *is_model_visible()* method to read the visibility informations of an object on the 3D scene). These functions may be used by other parts of the webpage to display 3D information contained (or created) in the viewer.

- A series of function hooks to track the events generated by the viewer (e.g. the *on_change_model_visibility()* method to check if an object on the 3D scene changes its visibility state). These handlers may be exploited to check whether an animation has terminated, the user has clicked over an object, or an instance has reached a certain state.

Through these elements a publishing system can make available different strategies to integrate 3D and Web content, for example proposing **hyperlink-based schemes** that let the webpage to manage the 3D scene from outside the CANVAS. Exploiting this technique the aforementioned functions would be called by HTML elements, making it possible to control the viewer behavior or composition from the webpage content.

For instance, in the case of a single model 3D scene associated to external textual information, could be possible to connect the action of clicking a text area that describes a detail of the presented 3D element, to the position change of this latter, with the aim to show the described detail. Or even, in the case of a 3D models gallery published in a webpage also containing pictures of these models, may be possible to link the mouse-over event, triggered when the mouse pointer enters one of these pictures, to the retrieve of the related 3D model from the gallery.

Among the solutions allowing this level of multi media integration there is the already introduced ShareMy3D [AH15] system. It provides a basic set of JavaScript functions (well documented, with running examples on its official websites), that are able to drive the 3D scene through webpage components external to the CANVAS area. But obviously, this kind of hyperlink-based scheme is not the only way to integrate 3D content and other media in a webpage.

Nowadays the dominant approach, alternative (but not concurrent) to the one previously presented, pursues an inverse path. It consists in bringing the other media into the 3D scene, rather than migrating the viewer commands outside of it. This technique makes use of the Web3D *immersive interfaces* (introduced in §2.4.3), designed to embed or to superimpose graphical elements directly as floating elements in the 3D scene. As stated by Jankowski et al. [JD12a], in these interfaces 3D graphics is the main information carrier, able to provide to the content-consumer both the freedom to navigate the 3D environment and the hypertextual-like data (directly immersed into it).

The implementation of this approach for multimedia integration rely on the possibility to attach, at the scene level, information to 3D environment and 3D objects, a mechanism that is the base of the ***annotations/hot-spots*** systems, features currently offered by many Web3D solutions (A-Frame [Moz15], BabylonJS [CR13], Blend4Web [Tri14], PlayCanvas [EEKI+11], etc.). These systems provide notes to "stick" on 3D models, useful for adding information to a specific part of them (see Figure 2.10). Each note usually has a position, a camera position, a title, a (multi-) media content, and, sometimes, also an order in a numerical list. Final users can view the attached information by interactively recalling these notes during navigation. In response to this action, the additional content is presented (typically in screen space) adjacent to the associated object.

The connection between multimedia data and 3D scene can be achieve adopting different strategies. One of them consists in placing on the 3D environment *labels* (or other hyperlink-based 2D elements) connected to objects of interest via placeholders (usually anchors to specific 3D points in scene space). Generally these clickable components are HTML (or HTML derived). This allows Web3D solutions to exploit, without much effort, the native interaction methods provided by the markup language. Another possibility is to transform the 3D objects on the scene in *clickable instances*. This strategy does not need placeholders but, for driving the user interaction, it requires that the 3D system provides a set of event handlers specifically related to the 3D environment (this feature is usually provided just by the more complete Web3D solutions). Implementations of the *labels* and *clickable instances* approaches can be respectively found in: Cl3ver [IN13], a Software as a Service (SaaS) to edit and display 3D content online (see Fig. 2.10); and WhitestormJS [Buz15], a JavaScript framework for simplifying 3D Web publication deployment (it adds physics and post-effects to the Three.js [Cab10] technology).

One of the biggest weak point of the multimedia integration methods introduced in this section concerns their usability, intended as easiness of setup in the publishing

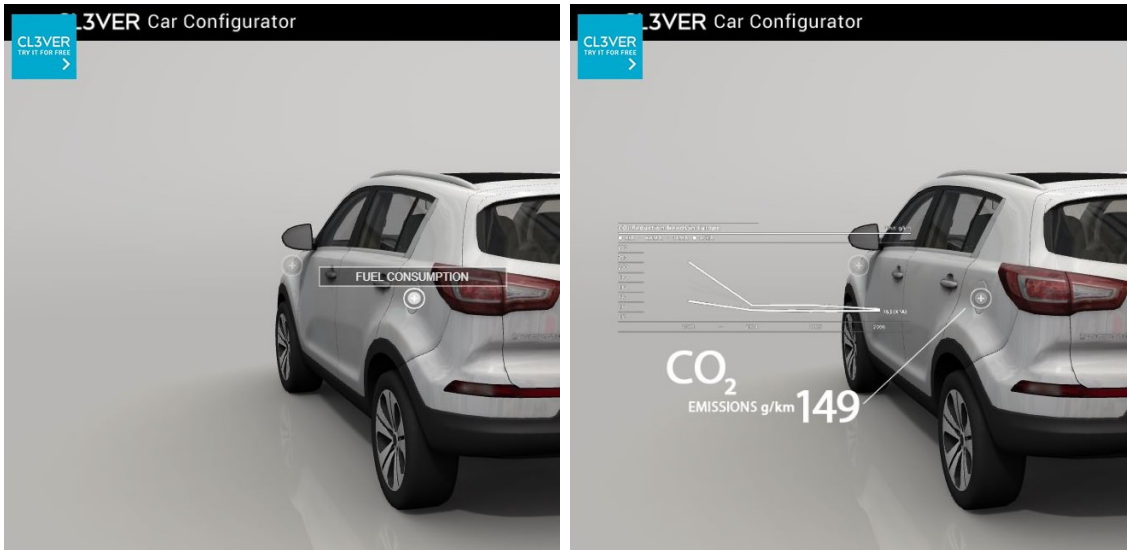

Figure 2.10: Multimedia integration via annotation system in the Cl3ver [IN13] solution. In this example clicking on the circular 2D graphical placeholder anchored to a specific scene space 3D point of the scene (left) the final user can get the additional information presented in screen space as superimposed HTML element (right).

creation stage. All the presented methods require the content creator to spend a significant amount of time and effort, to upload, link and configure the additional layers of information. While this may be irrelevant for a professional content creator, a casual one may find this work overwhelming.

Many systems, trying to get coexisting multimedia integration and democratization of use (both pillars of modern Web3D), have resorted to specific ***authoring tools***, able to interactively guide the connections between 3D and other media step by step. This approach seems to perfectly meet the intent, and has been fruitfully implemented by systems like the already mentioned Autodesk ReMake [Aut15] or Sketchfab [Ske14]. In these solutions the authoring tool covers all the areas related to the Web publishing. However, the implementation and maintenance of an authoring system are cumbersome tasks, and require a *server infrastructure* able to run them, narrowing the feasibility of this approach to the solutions deployed as services (or, in alternative, to stand-alone offline software equipped with a wizard).

How to support these needs remains an open issue for all the other self-publishing or serverless approaches. But, more in general, the 3D layer informative enrichment in his whole is still today a very active research field. Indeed, in the last years, several works have been addressed to improve the technology behind interconnection systems between 3D and other media (not always specifically aimed to the Web world). Russell at al. [RMBB+13], for instance, have explored the possibilities provided by automatic annotations,

Lehmann and Döllner [LD13] have tried to increase the potentiality of labeling 3D content in virtual worlds (not necessarily online), while more recently Seo et al. [SYK15] have proposed a method to enhance the annotations perception by rendering them with a 3D layout context and a camera perspective common to the related object.

2.4.5 Publishing Context

The characterization discussed so far has been mostly based on specific technical choices of the visualization system, considering the published data as the core of the Web3D environment. Nevertheless, to get a complete overview of the such ecosystem it is necessary to take a step backward, and to look at the online *publishing* phase. This section addresses the main aspects of the publication process, including:

- the publication modalities available (coding aimed at, node-based, full graphical user interface);
- the publishing purposes (pure visualization, assisted content creation, collaborative editing);
- the level of experience (specialized analysis, interactive presentation, social sharing).

One of the preliminary discriminating points of the 3D Web publishing process is related to the **target content creator** expertise level. This results in software systems addressed to naïve users and others designed for skilled developers. At publishing time, it is possible to choose among different development styles, that range from *pure coding* proposals to *GUI/wizard approaches*, passing through *node-based editing* and *block programming*.

The list of systems where the viewer is created by coding, includes a wide set of softwares, like for instance MathBox [Wit12], a library for rendering presentation-quality math diagrams in the browser, or LayaAir [CH15], a dedicated open-source API for games and multimedia routines modules. These tools are usually aimed to develop more complex or highly specialized applications (like are online games or analytical tools). Such solutions in one hand allow for a higher customization and a greater control of the resulting publication, but on the other hand require knowledge of one or more coding languages (JavaScript, HTLM, TypeScript, etc.) and, for this reason, are generally targeted at specialized/professionals users.

Less demanding, from the creation/setup point of view, are the node-based solutions, like for instance Goo Create [Goo12], a complete 3D authoring platform for cloud-based 3D content creation, or CopperLicht [Amb10], an open source 3D library equipped with a full 3D world authoring editor. These systems do not require writing code (even if often include sections where this is still possible) but provide the content creator with a drag'n'drop interface for visual programming,

where the scene, its behavior and interaction is defined by assembling and configuring predefined elements.

Finally, there are solutions which do not require the use of a programming environment, like for instance Koru [Box16], an authoring software that helps to prepare 3D models and to export the result online, or Kokowa [ZFF15], a publishing platform for non-programmers, aimed to create and share 3D and VR spaces. In these tools, a GUI is used as a wizard setup to drive the publishing step by step (typical of SaaS systems). These approaches do not require any kind of coding, resulting ideal for naïve users; the drawback is that they are more general purpose and do not allow to fully personalize the published viewer.

An alternative categorization of the Web3D systems can be based on the ***publication aim*** they promote, intended as basic purpose/target of the publishing. This means differentiating, for example, the solutions more focused towards *Digital Content Creation* (DCC) from others specifically addressed to *pure visualization*, characterizing all of them depending on the degree of DCC support provided (intended as real-time object/scene editing possibilities).

Indeed, for a number of systems the presentation feature is predominant with respect to the digital content design component (very basic and often only related to the customization of the scene). Kubity [Kub13], a cloud-based 3D player mainly aimed to enable final users to experience models in AR/VR online, could be a perfect example of that.

In many other cases, more technical and specialized, the ratio between presentation and creation support result inverted. In these latter solutions the Web viewer has the primary purpose to support the content creation process, whose final result could even not to be a public online publication (like in the case of the MeshLabJS [Vis14b] editing platform). Such systems are often addressed to technical users and usually provide support for collaborative DCC. A brilliant example of one of these was the Lagoa [DC11] platform (the cloud-based system for online CAD designing shown in Figure 2.11, whose professionalism has been recently acquired by Autodesk Fusion 360 [Aut13]).

Of course, a full set of hybrid systems exists between these two extremes, where the proportion of the ingredients is more balanced. This group includes solutions focused on Web visualization, but at the same time able to provide technical features oriented to assist the digital content creation, for instance supporting modeling and animation (the already introduced Clara.io [Exo13] platform could be a representative example for this class).

An additional categorization of modern Web3D software may be driven by their specialization. Indeed, nowadays these systems are more and more designed to satisfy specific needs, or focusing on peculiar goals, that go beyond the general-purpose visualization. This can lead to a characterization based on the *target uses* of the analyzed platforms, that may be applied not only to the basic aims of a publishing solution, but also to its ultimate target purposes, intended as type of ***supported experience***.

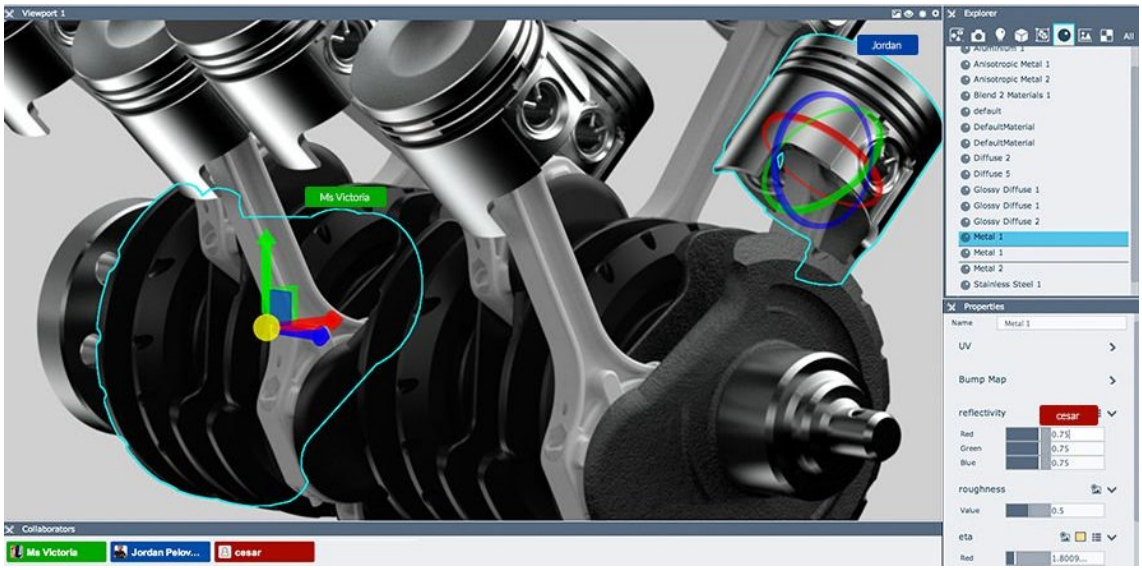

Figure 2.11: Collaborative DCC implementation in the Lagoa [DC11] Web3D system. This cloud-based solution aimed to CAD design guaranteed a performant real-time multi-user synchronized scene editing.

For instance, as already stated in §2.4.3, solutions aimed to specialized technical fields generally provide *specific presentation or analysis components*. Systems aimed to *Geographic Information System* (GIS) visualization (like Cesium [Ces11], a JavaScript library for 3D globes and maps creation) often assist the development with a set of features (geo vector formats, map projections, Columbus view, large distances and coordinates handling) hard to find in other contexts. Instead, a solution designed for creating games and similar interactive 3D applications, such as Unity3D [Uni05] or the Unreal Engine [Epi14] (another multi-platform game engine equipped with a WebGL exporter), usually offers very different components, like for instance integrated physics, networking, or even spatial audio.

In §2.4.3 we have also seen that characterizing features connected to specialized types of experiences may be related not only to peculiar presentation or analysis elements, but also to *distinctive interaction paradigms*. For example, a Web3D solution designed to present architectural models will exploit the first person navigation mode, to let the user walk through the 3D scene. The already introduced Archilogic [Arc14] system can be taken as example of that category.

Interpreting the concept of supported experience in the broadest sense, also side elements of a publishing platform can concur to integrate this level of characterization. For instance, solutions aimed to the online sharing of 3D creations often propose social media related tools (with linking or embedding capabilities) and virtual spaces (models galleries or showcases) for supporting the (profiled) users. These *"community-oriented" features*, external to the rendering

context, even if seem to be apparently disconnected from it, actually represent a characterizing factor as important as the other in the virtual Web environment, able to make the difference in the choice (and in also in the success) of a Web3D platform (in this sense the Sketchfab [Ske14] case can be considered a representative example).

A final high-level classification of the available solutions can be drawn out considering eventual ***costs and distribution licenses*** of the reviewed systems. Like for any other software product, also in this case the usage terms and costs are important characterizing factors.

Among these costs, the choice between a *client-based* and a *server-based* platform can influence in a relevant way the costs of a specific solution. As we have already seen in §2.4.1, the need for server-based capabilities (storage, computation, user management, database support) of some of the existing solutions often leads to non-trivial expenses (for content-creators interested in self publishing), or to the payment of a subscription (for final users of commercial services). From the point of view of users, current trends generally contemplate the combinations of a basic services for free (with terms restrictions) and more specialized features which become optional and fee-based (with less or no term restrictions). The PlayCanvas [EEKI+11] system is just one of many adopting this policy.

Some on-line platforms are offered free-of-charge, even if they require high maintenance cost, because they may be used as a "beachhead" to promote ancillary products or other paid services. As an example, this is the case of Thingiverse [SP11] (created and maintained by a 3D printer manufacturer to provide the users with usable content, thus promoting the sales of 3D printers) or ReMake [Aut15] (created and maintained by Autodesk as a way to introduce their software to possible clients).

Fortunately, thanks to its wide landscape, the Web3D field has solutions offering any kind of available possibility, with solutions *open-source* (e.g. CopperLicht [Amb10]), *freemium* (e.g. Sketchfab [Ske14]), or *commercial* (e.g. ShareMy3D [AH15]), and licenses like *Apache* (e.g. BabylonJS [CR13]), *MIT* (e.g. A-Frame [Moz15]), or *GPL* (e.g. Blend4Web [Tri14]).

2.5 Discussion

As already discussed in §2.3, a single, organic description of the Web3D panorama is complex, due to the necessity of building a low- to high-level categorization of the available solutions. For this reason, we tried to look at this heterogeneous landscape from two different points of view. In the previous section, we presented an analysis of the different features, organizing them in functional macro-classes. Conversely, in this section we try to work orthogonally, synthesizing the topics introduced in §2.4 into a general reference scheme, and use it to outline the profile of the various existing approaches/systems, connecting each of them with their

characterizing features. These same features will be finally projected on a representative number of application fields which more than others benefit from Web3D technologies, in order to assess the ideal approach for each of them and also to enrich our classification pattern.

2.5.1 Classification

In order to provide a map enabling the assessment of the Web3D solutions introduced so far, we propose a classification based on the features previously selected. This reference scheme, connecting these features to fully qualified products, is also useful to evaluate how characterizing these aspects could be for real world 3D Web systems. Thus, our classification is primarily based on the characterizing points introduced in §2.3.2, represented in a table able to provide an overview of the amount of support given to each of them by the reviewed systems.

Ideally, we could have mapped each feature on each one of the Web3D solutions reviewed so far, but we decided not to follow this avenue mainly because of the difficulty to represent in a visually accessible table the large number of crossings eventually obtained (approximately one hundred, since we should intersect around twenty features with the around fifty systems introduced).

Moreover, also the considerations about the effective usefulness to explicitly refer to existing solutions have led to avoid this kind of representation, condemned to become quickly obsolete in a technological environment in rapid evolution such as Web3D, where new solutions are systematically released and old ones are fast to disappear. A brilliant example of that could be represented by the Autodesk ReMake [Aut15] software, moved by the company to the new Autodesk Recap Photo just during this thesis drafting (after it has already been moved from the Memento test project in 2016).

Therefore, we decided to proceed in a different way, and, exploiting the schematic representation proposed in Figure 2.2 and in Figure 2.4, we decided to simplify and synthesize the reviewed approaches and features harmonizing them in a representative number of reference classes (LIB/LIB++, TOOL/TOOLKIT/FRAMEWORK, APP) and characterizing levels (DATA, SCENE, INTERACTION, INTEGRATION, PUBLICATION). Table 2.1 shows the result of this classification, giving an overview of the surveyed software based on the aforementioned criteria. The scheme outlines which publishing feature or technique is supported by which application domain. The amount of support is expressed as a range between a couple of values selected among: no [-], low [*], medium [**], and full [***] support.

Presenting the options that a practitioner has at hand when 3D content for the Web has to be created, the table also provides an implicit understanding on which level of complexity/difficulty of use is needed to obtain high quality results.

Analyzing the proposed scheme it is also possible to glimpse some interesting trends. In particular, scrolling through the table from the features point of view

Table 2.1: Web3D characterizing features mapped on the various publishing approaches. The solutions are classified according to a representative selection of the criteria given in §2.3.2 (bullet list). Software systems and features are grouped (respectively) as in Figure 2.2 and 2.4 in order to provide quick overview of methodologies and options that a content creator can access. Each cell shows the amount of support given by a group of systems to every single features level, expressed as a range between a min and a max value. The elements are sorted by supported features in order to show potential trends. The gray cells represent the transversal behavior of stand-alone end-products (full GUI applications mainly working locally, characterized by access and customization features similar to low/middle level approaches).

		LIB, LIB++	**TOOL, TOOLKIT, FRAMEWORK**	**APP**
DATA	Representation Types/Schemes Encoding/Transmission Formats Pre-processing Possibilities	**/***	*/***	-/**
SCENE	Spatial/Logical Definition Modes Appearance Customizations Features Animations Possibilities	**/***	*/***	-/**
INTERACTION	Tasks and Paradigms Interface-based Components Specialized I/O Modalities	*/***	*/***	*/**
INTEGRATION	Hyperlink-based Schemes Informative Enrichment Tools Authoring Elements	-/**	*/**	*/***
PUBLICATION	Content Creation Facilities Real-time DCC Modalities Specialized High-Level Elements	-/*	*/**	**/***

(row-major order from top to bottom), it attests that low level functionalities (mostly DATA and SCENE related) are usually better handled by libraries, APIs, or local solutions, while INTEGRATION/PUBLICATION features are generally more broadly supported by higher level systems (mainly thanks to GUIs or authoring interfaces). At the same time, moving to the systems point of view (column-major order from left to right), the table points out that, while the LIB/LIB++ and TOOL/TOOLKIT/FRAMEWORK supporting curve decreases its values going from top to bottom (i.e. from low level to high level features), the APP trend follows the opposite principle, significantly increasing the support amount given to each characterizing level shifting from from top to bottom.

Finally, this schematic visualization is important also to confirm the point stated in §2.3.1 concerning the issues in clearly classifying all the wide range of solutions in a universal fixed scheme. Indeed, seeking to represent in the table, for instance, stand-alone softwares aimed to a two steps 3D Web publishing (local model/scene processing and following online exportation), we would get an unusual characterization. The level of support for this class of solutions (essentially full GUI applications providing customization possibilities equal to middle level systems and data/scene access comparable to low level approaches), in fact could be ported in the proposed scheme only following a transversal representation. The (gray) colored cells in the table just serve to highlights such particular case, in which we can convey all these commercial end-products that have to be downloaded and installed, like for example Unity3D [Uni05] or Autodesk ReMake [Aut15].

2.5.2 Application Fields

In order to specialize the reference scheme introduced in §2.5.1 we apply its features to some representative application fields. In addition to provide an overview of the systems best fitting each specific field of application, we also want to define the full set of features that content creators could have at hand while creating specialized 3D Web content. We notice that for some of these fields the solutions tend to be all of the same type, while other fields present more heterogeneous set of solutions.

Cultural Heritage

Cultural Heritage applications, and related solutions, are mainly related to publication of content (with some degree of personalization). Usually the 3D data involved in this kind of publishing are high resolution 3D models coming from real world acquisitions (digitized with photogrammetry/structure from motion approaches or active scanning devices). Handling online these dataset (big size, huge complexity) requires to adopt *multi-resolution representations* and *performant data transfer formats* (so, a *pre-processing* phase is usually needed). Since often the 3D model to be published represent copyrighted objects, the *IPR*

protection and the *infrastructures security* are relevant in CH systems that offer cloud services. Other central elements concern the possibility to *integrate informative content* to the 3D layer, and to *support innovative I/O devices* (these two aspects are of fundamental importance while building virtual museums). *Camera animations* can also be really useful, while *model animations* are not a strict requirement. Effective interactive 3D scene *inspection* and *navigation* are both mandatory features (for architectures and artworks 3D models). Conversely, *scene customization and editing* tools are not so important, due to the need to convey the proper message and to pursue high fidelity in the visualization (they can also be dangerous since they could produce model appearance modifications).

CH is characterized by an heterogeneous set of application cases and requirements, and still by a low economic value (since most of the applications are implemented on a low budget: as an application domain it does not attract much interest from commercial companies and the commercial tools specifically designed for this application are still a small set. For this reason, the CH community is very often using tools developed for other purposes and domains, such as games or animation (immediate examples are Unity3D [Uni05] and Unreal Engine [Epi14]). This does not prevent that systems not specifically designed for CH (such as Sketchfab [Ske14] or Autodesk ReMake [Aut15]) actually fit quite well the CH application needs (e.g. Sketchfab has a dedicated section only for supporting Web publishing of Cultural institutions). Some academic solutions, like Potree [Sch13], even if don't explicitly refer to CH, at the end are mostly used to publish CH 3D models. However, also ad-hoc CH-oriented solutions have been proposed: mostly ignited by cultural or academic institutions, unfortunately they often are "blackbox" systems of restricted use (like in the case of 3D viewers by the Petrie [Uni09] or the Smithsonian [Smi11]).

Biomedical

The biomedical applications are typically focused on visualization, enabling the rendering of particle systems (by *ray-casting*) and volumes (by *ray-marching*). This could direct the choice of the proper publishing platform on systems able to profitably handle these dataset on the Web (even if, as we have seen, triangle-based techniques could be also exploited). Due to the technical nature of these publications, the presence of *specialized analytical tools* (enabling visual and numerical data analysis, such as unit cell operations, computation of distances and angles, torsion angle measurements, etc.) plays a key role, as well as the possibility to use *interaction paradigms* tailored on inspecting this specific 3D content. For the same reason Web3D solutions providing interactive (or collaborative) *annotation systems* may be preferable. Finally, *animations* features results also extremely useful in biomedical presentations, especially when they are characterized by didactic and dissemination purposes.

Among the systems presented so far (besides the more research-oriented

proposals, like X3DMMS [ZCGC11] and MEDX3DOM [Con12]), it worth to mention a couple of open-source projects, like X Toolkit [XTK12] (framework for visualizing and interacting with medical imaging data, it provides a simple API offering native support for neuroimaging file formats), and JSmol [Jmo13] (molecular viewer for chemical structures in 3D with features for molecules, crystals, materials, and biomolecules).

GIS, Maps and Architecture

This publishing field covers an heterogeneous set of Web3D systems, all them somehow dedicated to the creation of maps-based applications (at different levels of detail and complexity). Almost all these approaches require to handle dataset structured as *LoD tree* for progressive view-dependent refinement, that probably need to be *georeferenced* (particularly in Web GIS solutions). Maps apps often make extensive use of *geometry instancing* for repeated object on the 3D scene, and not rarely exploit *camera animations* supporting "spatial" storytelling (the recent Voyager function in the popular Google Earth [Goo11a] application brightly implements this feature). Of course, the *navigation component* (usually first-person/walk-through) is equally important enabling the interactive exploration of these dataset (especially for architectural models). Finally, also this domain requires a peculiar set of *specific presentation/analysis features*, like for instance: individual object picking, atmospheric elements drawing, precision handling of large view spaces (avoiding z-fighting) and large world coordinates (avoiding jitter), or time-line controls for the simulation of time-varying phenomena.

Nowadays, GIS visualization and analysis on the Web can be exploited using a number of interactive systems specialized on vast volumes of geospatial data and able to provide vector graphics, surface models and 3D buildings. We have both commercial solutions, such as GeoWeb3D [Geo12] and GeoBrowser 3D [Gra16], as well as open-source resources, like OpenWebGlobe SDK [Uni11] and the already mentioned Cesium [Ces11].

Maps-based Web3D approaches can be aimed not just to geographical visualization (like the popular Google Maps API [Goo11b]), but also to visual explanatory generic-data analysis, like in the case of Deck.gl [Ube15] (complex visualizations of large datasets rendered as stack of visual layers) or Seerene [HB15] (source code interactive analysis software maps-based).

Maps driven visualization is also useful platforms addressed to architecture, such as the systems Archilogic [Arc14] and 3D Wayfinder [3D 12a], both specialized in floor plans and 3D buildings interiors and exteriors exploration.

CAD

CAD solutions span a wide space, going from simple to professional systems. This mainly depends on the complexity of the operations and task required. Web3D publishing solutions reflect this point. Models exploited in this application field are 3D meshes, usually characterized by a relative structural simplicity (in term of number of triangles/faces), since they are generated (and serve) in design and modeling process. For this reason, and mostly for need of ensuring coherence in the geometric editing, they generally rely on *single-resolution formats*. A central feature for this kind of application is for sure the *support to DCC*, intended both as *specialized tools* (kinematic assembly, geometric constrains, distances and angles offset, etc.), and higher level features, like *collaborative editing* and *version control*. For more technical uses (for instance in industrial pipelines aimed to photorealistic publication) it is fundamental also to be able to steer the 3D object/scene *appearance editing* (shaders, materials, lighting, etc.), as well as the possibility to have access to the proper *authoring tools* able to drive these specialized operations.

The landscape of solutions addressed to CAD design includes systems ranging from unskilled users (e.g. Autodesk Tinkercad [Aut11]) to professionals (e.g. Lagoa [DC11]). Some of the proposed approaches are more oriented to editing/modelling (e.g.Autodesk ReMake [Aut15]) while some others are more focused on the publishing stage (e.g. Koru [Box16]). Commercial solutions, like ThreeKit [Hou15], and free softwares, like OpenJsCad.org [MJM13] (JavaScript Web interface for programmatic modeling), are both provided.

3D Printing

3D printing -oriented Web applications are, generally, easy to use platforms, where the aim of the publication is mostly focused in supporting user-uploaded sharing (or selling) of printable content.

These systems, mostly implemented as services, provide *hosting services* and a number of related high-level features (like *IPR management* strategies, *users profiling*, *community-oriented tools*). Since printable objects must be simple single item, Web3D printing solutions often propose bare-bone viewers aimed to *pure visualization* of a *fixed-resolution* model.

3D printing models have gained momentum in the last years. Due to this explosion of interest, have been released a lot of SaaS sharing platforms where final users can interactively visualize this content before to download it. Generally these system are marketplaces where upload and sell 3D object related files (like Pinshape [MSY13]), but there exist solutions implemented as open virtual space where to share them for free (like MyMiniFactory [MyM13] or Thingiverse [SP11]). The Threeding [Thr13] platform in addiction to allow to share 3D printing models and files (for free and paid), also provide a service for on-demand 3D printing (for users without 3D printer). The Shapeways [Sha13] startup instead is completely

based on the concept of printing (and selling) service, giving to the users the possibility to upload 3D printable files, and printing the out-coming objects (in over 55 materials and finishes) for themselves or for others. However, not only Web service platforms are characterized by 3D printing features. Online CAD systems, for instance, often provide simplified creation features just to make possible to print the designed models (Autodesk Tinkercad [Aut11], Leopoly [Leo15], and BlocksCAD [Blo17] belong to this category). Sometimes also less specialized solutions may provide support to 3D printers standard (STL) file formats and to 3D printing models preparing, like for instance Autodesk ReMake [Aut15] does.

Games

Web3D solutions aimed to games developing are generally systems able to handle elaborated 3D scenes made of a large number of modeled geometries. For that reason they often provide features focused to *complex scene composition* (geometry hierarchically instancing). Particular care is addressed to components for customizing *scene appearance* (rendering/shading processes), *animations* (both cameras and models) and *exploration* (mainly ambients navigations). Of course, building a game experience also play a central role the possibility to access to *specialized features* like physics, networking, and audio control.

Several solutions can be exploited or adapted to Web3D games developing. Almost all the typologies of approaches are available: from low level engines (like PhiloGL [Bel11] or KickJS [NJ11]) to GUI based applications (like Goo Create [Goo12] or CopperLicht [Amb10]), from more general systems (like BabylonJS [CR13] or Blend4Web [Tri14]) to highly specialized softwares (like PlayCanvas [EEKI+11] or LayaAir [CH15]). Even if in this specific field the most popular solutions remain systems not specifically designed for Web3D, like Unity3D [Uni05] or the Unreal Engine [Epi14] (both the softwares are just equipped with a WebGL exporter), is easier to find Web targeted applications, like for instance Turbulenz [Tur09], a modular 2D/3D framework focused on HTML5 game development for desktops and mobile devices, or Voxel.js [Vox13], an game building toolkit that make it easier to create 3D voxel Minecraft-style games.

Chapter 3

3DHOP: 3D Heritage Online Presenter

As we have seen, the adoption of WebGL has, at long last, made the Web3D steer away from the trap of proprietary heavy-weight plugins, facilitating the development and the spread of Web publishing solutions, and also ensuring a noteworthy thrust for the the online deployment of 3D content in many contexts formerly forgotten.

Looking at the wide variety of features and approaches presented in Chapter 2, it may seems that, due to this explosion, every issue has already been covered by existing solutions. Conversely, at a closer inspection, it is possible to note that the rapid development of Web3D has left out many niches for specialist and technical users, and has restricted some of the analyzed features only to a specific field, neglecting others. From this point of view, many empty spots can be found in each one of the previously analyzed areas, and some "missing links" appear obvious.

At *data level* for instance, the efforts made so far have been mostly focused on mainstream (i.e. entertainment oriented) geometric structures, that are often characterized by low-poly textured structured content, failing to address a different kind of geometry, coming from digitization and survey, characterized by high-resolution and unstructured data. Moreover, the application of efficient LOD/multi-resolution schemes, that could be able to cope with these other geometries, mostly rely on closed/proprietary formats that often require unknown/uncontrolled data transformations. At *scene level* instead, the existing solutions, influenced by the declarative/imperative dichotomy, have finished for clustering in Web friendly systems (simple to use but limited in customizations) or CG aimed software (highly programmable but addressed to professional content creators), forgetting to explore more flexible and scalable solutions. At *interaction level* the main issues seem to be related to the specialization of the available approaches, able to provide, at the same time, ready-made paradigms too much tailored on a specific type of 3D content and barely customizable, but also user interfaces poorly specialized on the applicative domain. Concerning *integration level* instead, the trend to mimic offline applications has mostly led to solutions at

best focused in bringing basic media connections into "dumb" 3D viewers, generally disconnected by the page logic and extremely deficient in exploiting the hyperlink-based possibilities and in providing developers with Web style functions/events handler. Finally, analyzing the existing approaches at an higher abstraction level (*publishing level*), a relevant issue can be traced back to the searching for the best trade-off in content creator usability (meant as the inability of a single solution in covering a wide gamma of target user technical skills).

Although few of these gaps find partial solutions in the Web3D landscape, as seen in the previous chapter, it is also true that a single solution, able to effectively face all these issues, does not exist. Starting with this premise, and following the brief analysis of requirements just exposed, we have defined a research space with the aim to propose a functional solution able to provide sensible answers to the listed needs. Designing a viable implementation of this new tool, and then developing and testing it, we pursued the idea to create a flexible and efficient system, able to lead step by step to a fast and easy integration of 3D models into a webpage, expressly addressed to the online environment, and finally, tailored on one of those specific niches of technical users which are somehow far from the mainstream use of 3D data. Concerning this latter point, our choice has fallen upon the field of Cultural Heritage.

Indeed, thanks to its peculiarities, the Cultural Heritage field results particularly suitable to be exploited as concrete reference case. In this domain, digital 3D models are nowadays widespread and, beside their more "technical" use (documentation, restoration support, study and measurement) they are becoming very valuable in dissemination, teaching and presentation to the public. So, publishing 3D content online is more and more becoming an usual action for CH professionals. Moreover, since these 3D data are published on the Web for presenting museum collections, displaying virtual reconstruction of ancient sites, or showing connections between artifacts of interest, they also need to be strictly paired by (and communicate with) other multimedia information. But the most characterizing features for this domain probably concerns the need for a higher complexity of the 3D content to be handled online. While in many other applicative fields lower-resolution hand-modeled 3D objects may suffice, in the CH field the digital geometries, often digitized versions of real-world artifacts or environments, have a much higher resolution, both for keeping a scientific validity, and to convey to the public the correct information. Thus, by choosing Cultural Heritage as our target field, we had the possibility to face all these interesting problems:

- challenging situations for the management of the data level: high-resolution, complex, diverse data;
- the need for multi-level access: dissemination for final users, technical access to experts, and everything in between;
- conflicting requirements for simple-yet-flexible interaction methods and tools;

- different levels of publication needs: from the simple, single-object, "dumb" viewer, passing through the specialized viewer with customized interaction, to reach the large-scale publication support, for collections and institutions;

Finally, Cultural Heritage was a suitable field also because, thanks to the connections and collaboration of our research group, we had plenty of test cases and opportunities to validate our tool.

3.1 Overview

The final result of our work is 3DHOP (3D Heritage Online Presenter) [Vis14a, PCD+15a], a software framework specifically designed to simplify the creation of interactive visualization webpages displaying high resolution 3D models. It provides intuitive user interactions and also a deep integration between 3D data and the rest of the webpage elements. The most interesting characteristics of the 3DHOP framework are:

- the ability to work with extremely complex 3D meshes or point clouds (tens of million triangles/vertices), using a streaming-friendly multi-resolution scheme;
- the ease of use for developers, especially those with background in Web programming, thanks to the use of declarative-style scene creation and exposed JavaScript functions used to control the interaction;
- the availability of a number of basic building blocks for creating interactive visualizations, each one equipped with sensible defaults, providing a comprehensive documentation, and mostly, widely configurable.

Since 3DHOP is targeted on Cultural Heritage users group, most of the design choices address specific peculiarities of the CH domain, providing a series of features that are extremely relevant to this sector. However, CH is not the only application field dealing with very high resolution models and requiring a dense interconnection between those models and other data or media. In this sense, CH is a major domain of inspiration and assessment for our activity, but not the only application context for the 3DHOP technology.

3DHOP is based on the WebGL [Khr09] component of HTML5 and on the SpiderGL [Vis10] library (introduced in §2.1.2). This makes the framework extremely lightweight in terms of dependencies, and able to run on major browsers (Chrome, Firefox, Edge, Opera, Safari) and platforms (Windows, MacOS, Linux). It does not need plugins or additional components installed in the client, nor specialized servers.

3DHOP, started as an experimental tool for internal use, evolved during this PhD thesis into a fully qualified solution, released with an open source GPL license. The downloadable package, with documentation, a series of tutorials (how-to) and a gallery of examples is available at the project official website [Vis14a].

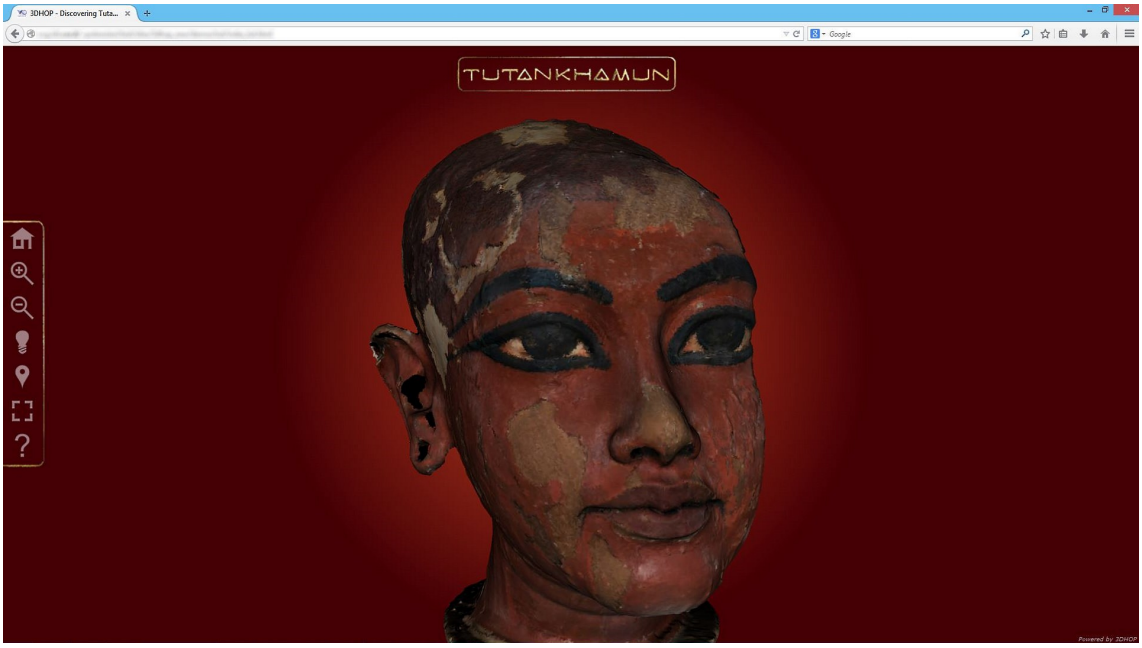

Figure 3.1: The *Tutankhamun* viewer: using 3DHOP [Vis14a] to publish on the Web a high resolution 3D model interactively explorable and multimedia connected (the artifact is linked to additional information through hot-spots).

Situating 3DHOP w.r.t to the State Of The Art

3DHOP is not a "silver bullet", able to support any possible application or visual communication project, but a framework designed to deal with specific needs.

It is an ideal tool to visualize high resolution single objects (especially dense models coming from 3D scanning, as shown in Figure 3.1) or, more in general, a simple scene composed of complex models. Conversely, 3DHOP is not suited to manage complex scenes made of low-poly objects (this is a common case when working with CAD, procedural or hand-modeled geometries), as the Web3D landscape already offers different solutions aimed at this niche.

3DHOP makes possible a fast deployment process when dealing with simple interaction mechanisms, making it a good choice for quickly creating interactive visualizations for a large collection of models. Additionally, 3DHOP integrates extremely well with the rest of the webpage, thanks to its exposed JavaScript functions. The ideal situation is having the logic of the visualization scheme in the page scripts, and using 3DHOP for the 3D visualization. Trying to build an interface directly in the 3D space using its components (i.e. clickable geometries used as buttons) is certainly possible, but the results do not scale well with the needed configuration work. In the following, three existing alternative solutions (already introduced in Chapter 2) are analyzed, in order to better stress similarities and differences.

Unity3D [Uni05] is one of the most common tools for displaying interactive 3D

content on the Web for CH applications, a de-facto standard in this specific field. It is natural, then, to compare 3DHOP with Unity3D. Unity3D is a full-fledged game engine, extremely powerful and complete, providing advanced rendering, sound, physics and a lot of pre-defined components and helpers. Unity3D supports the implementation of interactive visualizations holding the same level of graphics and interaction complexity as a modern video-game. It has a rapid development time when creating a simple visualization, but the complexity of use/development ramps up if it is necessary to employ the more complex interaction features. Moreover, Unity3D is not well suited to manage high resolution sampled geometry (except for terrains), while it is really good with hand-modeled geometry. Its streaming capabilities requires to pay a fee, and it also requires server-side computations. Finally, even if there are different ways to interconnect the 3D visualization with the webpage, this is one of the more complex features to set up in Unity3D, conversely to the otherwise user-friendliness of the tool. All these features make Unity3D somehow complementary to 3DHOP: the Web-integrated visualization of single, high-resolution artifacts finds in 3DHOP a better support, while the exploration of complex modeled scenes or even immersive environments are better managed in Unity3D.

Another popular solution, widely used even by the CH community, is Sketchfab [Ske14]. It is indeed extremely simple to use and offers fast online deployment of 3D models and data storage support. On the downside, Sketchfab has a limit on the geometrical complexity of the input models, making it difficult or impossible to upload 3D scanned models at full resolution (despite a blind data pre-processing, imposed by default). Moreover, the interaction with the 3D models is only partially configurable, making it difficult to tailor the interaction to the specific shape and characteristics of the model. Additionally, models are stored on a remote server, raising issues of data privacy and data property. Finally, being the result of a commercial initiative, the more advanced features, including the handling of more complex geometries, are available only in the Pro (paid) version.

X3DOM [Fra09b] is another development platform that gained a quite broad range of applications. As already stated, the X3DOM structure derives from a declarative approach and the definition of the scene is obtained through a *scene graph* concept and related commands. X3DOM has several points in common with 3DHOP: for instance, also X3DOM has a ready-to-use solution to handle medium-sized and large 3D datasets (over the time it introduced several binary container nodes, like Binary Geometry [BJFS12], Pop Geometry [LJBA13], and the more recent Shape Resource [LTBF14]). However, it is misleading to compare them directly, since X3DOM is more akin to programming language (based on the declarative paradigm), while 3DHOP is a set of configurable components (built using a different paradigm). X3DOM does implement default field values (following the specifications of X3D [Web04]), and it provides most of the basic components of 3DHOP. Nevertheless, even creating a simple visualization requires dealing with the complete setup of the rendering and interaction, making it

difficult for those with limited programming skills to obtain a step-by-step understanding (code for simple examples has been added to the official website only recently).

3.2 Design Choices of the 3DHOP Framework

3DHOP has been designed with the aim of being easily and functionally integrable in the Web ecosystem. Our core idea was to mimic the philosophy of those pre-made HTML/JavaScript components available online, for example for image slide-show, date or color picker, charts and graphs. These solutions can be simply plugged inside a webpage including some scripts and adding few lines of HTML, and used by just changing some variables; they interact with the rest of the webpage with a series of exposed JavaScript functions and events.

As well as these components also 3DHOP wants to be easy to use, especially from the point of view of people having a background in Web development, without compulsorily requiring solid knowledge in CG programming. Most Web developers have experience with similar components, and they are indeed extremely useful, given their quick startup, different configuration level (from a simple parameter change to advanced modding) and integration with the rest of the webpage. Just the integration feature, meant as possibility to merge the component and its content in the multimedia Web context, is another key factor able to impact more on the 3DHOP design choices.

Usability and integration of the presented framework have been pursued through specific design choices, particularly relevant in the definition of the 3DHOP inner structure: oriented toward a Web-friendly development paradigm; providing handlers aimed at linking the various webpage elements; possibly equipped with sensible default behavior; and finally, built to be exploited in heterogeneous applicative contexts and on different technical scenarios.

It is clear that directly using WebGL, or (better) relying on one of the higher level libraries, frameworks and paradigms analyzed in Chapter 2 (like for instance XML3D [Son10] or Three.js [Cab10]), it could be possible to create interactive presentation like the ones made with 3DHOP (or the entire 3DHOP tool) from scratch, but this would still be an “ad-hoc” effort. 3DHOP may be somehow restricting, with respect to a project-specific custom viewer, but we believe the ready-made components and behaviors and their reusable nature make it a valuable tool.

3.2.1 Declarative-Style Setup

As previously stated, two main development paradigms support the development of 3D Web applications: the *declarative* approach for the management of 3D content (e.g. endorsed by X3DOM), and the *imperative* approach (supported by

the introduction of WebGL in HTML5). The use of *declarative* 3D mimics the way the rest of the webpage is composed and managed: 3D entities (geometries, transformations, camera, animations, etc.) are declared and controlled as they are part of the DOM structure (like, for example, a DIV or an image). This approach makes things much simpler for people coming from the Web development side. Conversely, the *imperative* approach works in a way that is more similar to the implementation of stand-alone visualization software, by tapping into the capabilities of the graphics card using a more low-level programming. In most cases, it is like having the browser running an extremely powerful, stand-alone software, disconnected from the rest of the information available on the website.

If we apply a strong simplification of the current status, we may argue that the declarative approach is much easier for Web developers, not requiring specific knowledge on 3D programming, and provides seamless integration with the webpage, simplifying the development of interactive presentations of mixed data (3D/text/images/videos). On the other hand, the imperative approach enables the user to fully exploit the power of the graphic cards, at the cost of requiring much more effort in application implementation. Of course, things are never so simple, and, as we have already seen in §2.2.1, a lot of effort has been spent on both sides to reduce the separation of these two development paradigms. However, this dichotomy somehow is still holding and, depending on the personal background, it is quite easy to approach 3D Web applications design only considering one of the two paradigms, ignoring or misjudging the possibility offered by the other.

Our goal was to bridge the gap between these two worlds, by providing a framework that aims to combine the ease of use of the declarative style (to define the elements of the visualization and their properties), with the rendering power provided by low-level programming. We will describe in §3.3.2 how the creation of the scene follows a declarative style in 3DHOP, enabling a quick and intuitive (yet, highly customizable) deployment. At the same time, the core of the rendering exploits the experience matured in the field of CG programming (see §3.3.1).

3.2.2 Interconnection with the DOM

A quite common situation, especially when using imperative 3D systems, is the strong separation between the 3D visualization and the rest of the webpage. In most cases, the visualization tool is completely self-contained, not interacting with the elements of the page. This leads to difficulties in creating multimedia presentations, where an action on the webpage elements does affect the 3D visualization and vice-versa.

The system presented by Callieri at al. [CLDS13] was aimed at establishing a strong connection between what happens in the 3D viewer and the DOM elements, thus creating an integrated presentation context for different media. While succeeding in effectively connecting the imperative 3D to the DOM, the system was still limited by its specialization. It was possible, by changing some

configuration files, to display a different dataset, but the new object should be quite similar in terms of structure and semantics (the tool was tailored to CH artifacts with scenes carved on their surface, like, for example the Trajan column).

Conversely, 3DHOP has been designed to support the interconnection with the elements of the DOM in a more extended and configurable way. 3DHOP can work just as a "dumb" viewer (if the user does not configure any DOM interaction), but it offers many ways to interconnect the visualization to the rest of the webpage. It is possible to change the visibility of the different models (like in the example in Figure 3.2); select, read and animate the trackball position; activate hot-spots and detect clicks on the 3D models/hot-spots. Most of these features can be controlled just by invoking or by registering event-handling JavaScript functions provided in the framework. In this way, the Web developer has the complete freedom to integrate 3DHOP with the specific website logic.

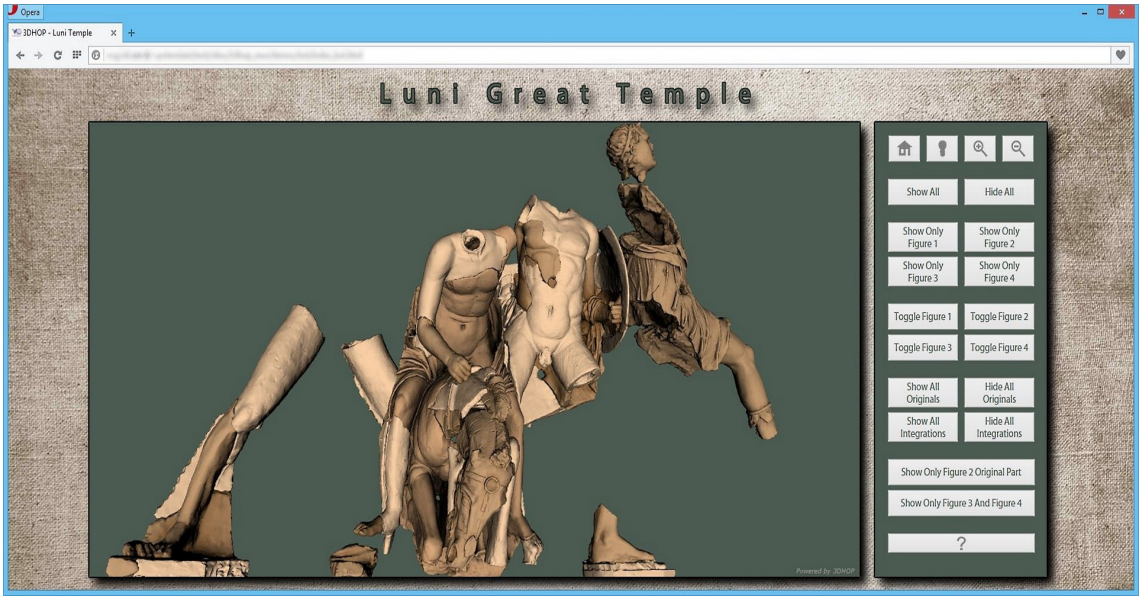

Figure 3.2: The *Luni Statues* viewer: in this example, four figures of the frieze of the Great Temple of Luni (Italy) are shown using 3DHOP [Vis14a]. Each statue has an original part and an integration; by using the visibility control, it is possible to control which subset of the pieces is shown.

3.2.3 Exhaustive Defaults and Level of Access

Another essential design choice of 3DHOP is to provide default behaviors, consistent with the common needs of our target community. Each component of the viewer is configurable, but it is never mandatory to modify/update each parameter. The developer may just change a single needed parameter, and rely on defaults for the rest of them. In a wide sense, we follow the *batteries included* philosophy of Python,

since we aim to simplify the life of the developer providing ready-to-use visualization components for online CH applications. In this way, our framework is much more accessible, and can be easily learned step by step (using the provided examples and the how-to resources). This also provides a fast startup when deploying new content (in many cases it is only necessary to do minor changes to the provided examples) and it is ideal to automate the creation of "3D galleries" when a large number of objects have to be presented, since the basic visualization can be easily created by a script. A completely unskilled developer may readily start using 3DHOP to visualize his own dataset by simply downloading one of the examples and changing the name of the 3D model file. Then, it will be easy to modify the parameters of existing elements to achieve more advanced results. A Web developer could approach the tool from another direction, by modifying the CSS/HTML to customize the graphic of the visualization. By using JavaScript, it will be then possible to connect the behavior of 3DHOP to the active elements of the webpage. A programmer with some skills in JavaScript and computer graphics may modify the trackball or try to add a new trackball to obtain a different interaction, or to customize the rendering by changing the shaders or the rendering of the scene. More expert developers can add new elements to the scene, setup new event hooks and heavily modify the viewer.

3.2.4 Online and Offline Deployment

While the 3DHOP framework has been designed for online applications, we also made possible its use on a local machine. Given its minimal interface, compatible with touch screens, and the ability to work without a dedicated server, 3DHOP is a good candidate for the creation of multimedia kiosks and interactive displays running on local machines inside a museum or an exposition. When deployed on the Web, 3DHOP does not require a dedicated server or server-side computation; some space on a Web-accessible server is enough to publish visualization webpages. This makes deployment easier also for institutions without complex IT infrastructure (like most museums); moreover, this self-publishing also avoids property and copyright issues (extremely important in the CH domain) related to the storage of restricted-access data to remote servers.

3.3 Inside the 3DHOP Framework

To harmonize the description of the features characterizing the proposed solution, and also to promote a easy comparison with the systems introduced in Chapter 2, once again we decided to resort to the schematic model proposed Figure 2.4.

The discussion will be so structured following the classic five macro-group pattern (DATA, SCENE, INTERACTION, INTEGRATION and PUBLISHING), as usual covering the full spectrum of peculiarities, ranging from low level to high

level functionalities.

3.3.1 Complex Data Management

As we have seen in §2.2.2 the efficient visualization of 3D data over the Web is mostly related to three key elements: poor computational resources, low network capabilities, and size of data to process. The latter point arise to dominant factor in the specialized context of CH, usually characterized by 3D content with relevant intrinsic complexity. For this reason, one of the most important features of 3DHOP is the ability to manage very high resolution 3D datasets by using a multi-resolution approach.

The multi-resolution approach ensures efficiency of both data transfer and rendering. Progressive schemes generally split the geometry into smaller chunks. For each chunk, multiple levels of detail are available. Transmission is *on demand*, requiring only to load and render the portions of the model strictly needed for the generation of the current view. While this approach is key to being able to render very large models at an interactive frame rate, of course it is also highly helpful with respect to the data transfer over a possibly slow network, since the data transferred will be divided into small chunks and only transferred when needed. The advantages of using this types of methods are the fast startup time and the reduced network load. The model is immediately available for the user to browse it, even though at a low resolution, and it is constantly improving its appearance as new data are progressively loaded. On the other hand, since refinement is generally driven by view-dependent criteria (observer position, orientation and distance from the 3D model), only the data really needed for the required navigation are transferred to the remote user.

The 3DHOP solution is based on a multi-resolution data structure described in [PD15, PD16], which allows the client to efficiently perform view-dependent visualization. Together with the low granularity of the multi-resolution this approach allows interactive visualization of large 3D models with no high bandwidth requirements (a 8 Mbit/s is sufficient for good interaction with huge models).

Smaller 3D models can also be managed using a single-resolution representation; currently, 3DHOP supports single-resolution models in PLY format [Geo14] (but more importers will be added as future work). In this case, the model file is fetched from the server as a whole and parsed by 3DHOP. This solution is ideal for small geometries (less than 1MB), generally used to give a context to higher-resolution entities or small modeled 3D meshes. The management of geometries, may they be multi-resolution or single-resolution, is completely transparent to the user.

Background: Offline Visualization of Huge 3D models

To handle large geometries over a network is certainly a relevant issue for some Web3D domain. However, the visualization of complex 3D content has been a trouble in Computer Graphics well before the possibility to have peculiar Web-based solutions.

Some of the issues related to 3D streaming had to be faced also in the offline context, and different seminal approaches have been proposed, like LOD based [RB93, FS93] methods, or the already mentioned solution proposed by Hoppe [Hop96] (game-changing, introduces the progressive refinement of the geometry during visualization). Following this work, a number of so-called multi-resolution and multi-triangulation solutions have been proposed. They mainly differ on the multi-resolution representation [SG05, CGG+05], on the support of color encoding [BGB+05], or on other aspects (a survey on these method was provided by Zhang [ZHXJ10]). Alternative research tracks are devoted to other types of data, like point clouds [WS06].

More recent work on this topic was devoted to the issue of data compression [LLD12] or to overcome the fact that multiresolution was mainly created for visualization and not for processing [GS12].

In general, the data structures used for offline visualization may be adapted to Web rendering, provided that they are compliant with its requirements (i.e. latency, decompression time). An alternative proposed solution was to still devote the rendering effort to a powerful server, and send to the user only a rendered image of the high resolution mesh [KTL+04].

Large Models Handling in 3DHOP

Displaying high resolution models on a Web browser is not just a matter of optimizing the rendering speed, but it also involves considering the loading time and network traffic caused by transferring a considerable amount of data over the network. While WebGL gives direct access to the GPU resources, how data is transferred from a remote server to the local GPU is up to the programmer. Loading a high resolution model in its entirety through the Web requires transferring a single chunk of data on the order of tens of megabytes: this is definitely impractical, especially if the user has to wait for this file transmission to end before seeing any visual result.

Starting from these premises, we decided to implement one of the state of the art multi-resolution schemes, called *Nexus* [Vis13], on top of the SpiderGL library [Vis10], obtaining very good performance. Nexus is a multi-resolution visualization library supporting interactive rendering of very large surface meshes. It belongs to the family of cluster based, view-dependent visualization algorithms. It employs a patch-based approach: the 3D model is split (according to a specific spatial strategy based on KD-trees) into patches; these initial patches represent

the highest level of detail of the multi-resolution representation. The number of triangles in each patch is halved, and adjacent patches are joined, in order to keep the number of triangles more or less uniform per patch. The different levels of detail are generated by iterating this process (bottom-up). The result is a tree structure containing each portion of the input object at multiple resolutions and, more importantly, the patches are organized and built to always match on common borders. This allows them to be assembled on-the-fly to build view-dependent representations at variable resolution.

At rendering time and based on the current view, the system decides which patches are better suited to represent the object given a target rendering speed and the maximum geometric error. Moreover, the batched structure allows for aggressive GPU optimization of the triangle patches, since the latter are encoded with triangle strips thus boosting GPU rendering performance.

At initial loading time, the "map" of the patch tree is downloaded, together with the lower-resolution patches. Then, depending on the view position, orientation and distance, the rendering algorithm decides which patches have to be fetched from the server to improve the current visualization, and queues a request. When each selected patch has been downloaded, the rendering is updated. The system continues this process of rendering-deciding-fetching-updating, trying to balance the amount of memory/data needed, the quality and speed of rendering and the network load.

All the data is contained in a single file. 3DHOP exploits the HTTP protocol capability to randomly access binary files to get specific data chunks inside each file, thus transferring only the needed portion of data. In this way, the viewer is able in a very short time to display a low-resolution version of the object, which is then progressively refined according to the user interaction, since the updates are view-dependent.

The conversion from a single-resolution 3D model to our multi resolution format is a one-time operation, done in a preprocessing phase. The 3DHOP user will convert its 3D assets using an executable (also open source, and included in the 3DHOP distribution). The obtained file is ready to be deployed on the Web server. It is important to note that our streamable multi-resolution encoding does not require server-side computation and resident data-streaming daemons. It is the client that automatically fetches data from the inside of the file, jumping from one location to another in the data structure.

Furthermore, as we have seen before, multi-resolution allows also some degree of data protection. Most institutions do not want their 3D data to be downloaded without permission. When using a multi-resolution encoding, the high resolution 3D model is never transmitted to the remote user in a single file but in a set of pieces encoded with a proprietary data structure. In this way, the malicious copy of the 3D data becomes quite complex and requires the design of ad-hoc procedures for downloading the whole geometric data and recombining them in the original model.

The first Nexus implementation [PD15] has been followed by a number of enhancements (mesh compression, point clouds support, textures handling), able

to further improve rendering possibilities and performances [PD16]. To give a practical demonstration of these capabilities we provide here some interesting data on a relevant selection of the introduced examples:

- *Tutankhamun* viewer (Figure 3.1): this example uses a multi-resolution Nexus model (compressed, textures embedded), for a total of 5 million triangles and 32 MB size (original fixed resolution models overall size: around 250 MB, textures included);
- *Capsella Samagher* example (Figure 3.6): this example uses 2 multi-resolution Nexus models (compressed, textures embedded), for a total of 10 million triangles and 142 MB size (original fixed resolution models overall size: around 270 MB, textures included);
- *Helm* viewer (Figure 3.5): this example uses 2 multi-resolution Nexus models (compressed, per vertex color), for a total of 15 million triangles and 50 MB size (original fixed resolution models overall size: around 305 MB);
- *Luni Statues* viewer (Figure 3.2): this example uses 8 multi-resolution Nexus models (compressed, per vertex color), for a total of 50 million triangles and 168 MB size (original fixed resolution models overall size: around 1 GB).

Comparing 3DHOP and Existing Solutions

We tested our rendering framework comparing it with the current state of art, in order to have tangible feedback about the effectiveness of our technical solution.

We chose to stream online the multi-resolution version of a relatively simple mesh, the Happy Buddha model (1M triangles, vertex color, 22 MBytes as binary .PLY file, previously used in similar comparison works [LCD13]), with some of the approaches previously mentioned. In these test we used a limited bandwidth internet access and, of course, the same hardware and software equipment (desktop PC equipped with Intel Dual Core i3-3220 CPU at 3.30 GHz, 8 GB RAM, NVidia GeForce GT 620 1 GB RAM, OS Windows 8.1 and Google Chrome Browser ver. 43.0.2357.124m). Since our framework uses a view dependent algorithm, for the sake of accuracy, it must be said that all the test have been run at Full HD screen resolution (1920x1080 pixels, aspect ratio 16:9), however, when handling around 1M triangles per model (as in the Happy Buddha case) our rendering system is indifferent to this parameter.

We compared the 3DHOP framework results against the Google WebGL-loader [Goo11c], the X3DOM binary POP Buffer Geometry [LJBA13] approach, the Sketchfab [Ske14] platform, and the Unity3D [Uni05] graphics engine, in order to have a wide selection of competitors, ranging from complete system solutions (X3DOM, Sketchfab and Unity3D) to stand-alone streaming services (WebGL-loader), from progressive mesh techniques (POP Buffer Geometry) to hybrid systems (WebGL-loader) and to more standard data

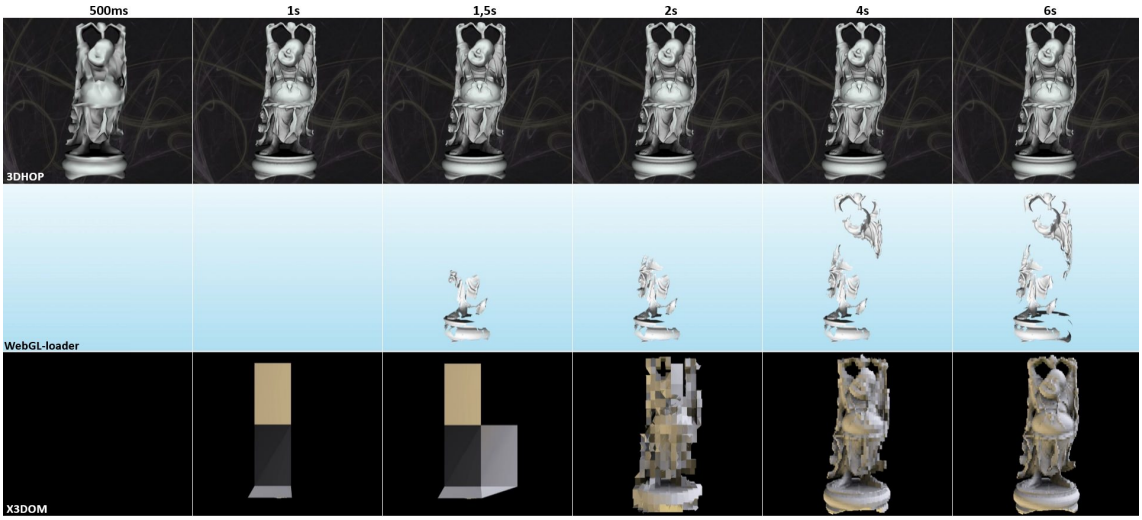

Figure 3.3: Comparative representation illustrating the Web rendering of a 1M triangle mesh on a 5 Mbit/s Internet access, using the 3DHOP [Vis14a] framework (first row), WebGL-loader [Goo11c] (central row) and X3DOM [Fra09b] binary POP Buffer Geometry (last row). All these test have been run on the same Web server to ensure equal conditions. From the left screenshots are taken respectively at 500ms, 1s, 1.5s, 2s, 4s and 6s after loading the webpage.

streaming procedures (Sketchfab and Unity3D), from completely free projects (WebGL-loader and X3DOM) to mixed solutions (Sketchfab and Unity3D).

The results of this comparison can be easily understood by observing the screenshots in Figure 3.3, representing the time-lapse visualization of the aforementioned approaches, respectively caught after 500ms, 1s, 1,5s, 2s, 4s and 6s from launching the loading of the Web pages. Under these conditions, with limited bandwidth (5 Mbit/s, typical 3G+ connection speed) and meshes with millions of triangles, it can be easily seen that 3DHOP (first row in Figure 3.3) is performing better with respect to the WebGL-loader algorithm (central row in Figure 3.3) and to the X3DOM POP Buffers system (last row in Figure 3.3). Readily after the webpage loading (500 ms), a rough version of the geometry is already visible, and can be used for user interaction.

It should be noted that the Sketchfab and Unity3D results do not appear in Figure 3.3; this because both Sketchfab and Unity3D viewers do not use (or just adopt simple) progressive loading schemes, and the model has to be nearly fully downloaded before it is visible. In both cases, the Happy Buddha model loaded after nearly 6 seconds from the Web page launch. It is clear that this gap with respect to progressive multi-resolution approaches is emphasized when the mesh size grows or the bandwidth decreases; on the other hand, it is also true that progressive multi-resolution systems may continue updating and streaming data also after the other systems will have transferred the whole model.

This eventuality can also be found by observing the data in Table 3.1. In this

	3DHOP	WebGL-loader	X3DOM
3,0 Mbit/s	**0,3** / **9,5**	2,0 / 19,4	0,6 / 44,5
5,0 Mbit/s	**0,2** / **4,8**	1,1 / 10,8	0,6 / 24,8
8,0 Mbit/s	**0,2** / **3,9**	0,7 / 6,8	0,6 / 15,2
20,0 Mbit/s	**0,2** / 3,7	0,3 / **2,7**	0,5 / 6,0
50,0 Mbit/s	**0,2** / 3,6	**0,2** / **1,1**	0,5 / 2,4

Table 3.1: Web rendering statistics for the Happy Buddha mesh (1M triangles) at different bandwidths (3, 5, 8, 20 and 50 Mbit/s), using 3DHOP [Vis14a] framework, WebGL-loader [Goo11c] and X3DOM [Fra09b] binary POP Buffer Geometry. Each table cell shows two average time (values in seconds): the first one concerning the start of the rendering (time that the user will wait before seeing anything), the second one related to the end of the rendering (whole 3D model drawn time). All these test have been run on the same Web server to ensure equal conditions (bold values represent the best performance in each individual case).

case the same Happy Buddha test previously presented, was performed at different bandwidths (ranging from 3 to 50 Mbit/s), this time taking into account the latency of the rendering (i.e. the time that the user will wait before seeing anything after running the webpage) and the end of the data streaming process (i.e. the time taken to render the higher resolution version of the model, given the starting point of view). Under the aforementioned conditions the table clearly shows indeed that on fast networks (20 or 50 Mbit/s) progressive multi-resolution approaches can employ a small amount of extra time to load the entire 3D model compared to the other approaches (an event that for our multi-resolution algorithm does not occur with lower bandwidths, when 3DHOP performs better than any other). However it should be stressed once again that our framework is able to provide to the final user a draft (but illustrative and ready to use) version of the whole 3D model practically with no waiting times (300ms in the worst case, with 3 Mbit/s Internet access), consistently out-performing other competitors in any situation (regarding this feature).

It is worth remembering that, to ensure equal conditions, all the tests in this section have been run on the same Web server, and, with respect to the data in Table 3.1, they have been obtained by averaging five different measurements per cell data. Finally, it is right to clarify that, in order to obtain results less dependent on external network interferences, during these tests the server and client ran on the same network infrastructure, but that the acquired results are comparable with those obtained with the client and server placed on two different network subsystems.

Currently, no quantitative test has been performed on mobile devices (since the mobile compatibility of 3DHOP is still not complete), but first results show that the performance of our framework will be good also on these systems (although the POP Buffer approach is extremely efficient on mobile devices due to the lack of decompression times).

A more detailed description and evaluation of the current version of the view-dependent multi-resolution engine can be found in a dedicated paper [PD16].

3.3.2 Declarative-Like Scene Setup

3DHOP has been designed to work with a few high resolution geometries, and not with really complex scenes made of hundreds of entities. However, it is still necessary to define a *scene* to initialize the viewer. The definition of the scene has been implemented in a declarative fashion. All the scene elements are declared as JavaScript JSON structures, with properties and values, and assembled into a comprehensive scene structure. This structure is then parsed by 3DHOP at initialization time to create the scene. We chose to use JSON because it is fairly easy to write and parse, it is human readable and easy to understand; XML would have been a good choice too, possibly a bit more verbose. With respect to a completely DOM-integrated approach, like XML3D, we are still somehow disconnected; the declarative approach is used to define the scene, which is an entity directly managed by the 3DHOP component, and all the interaction with the DOM passes through the 3DHOP viewer object, following the idea to create a self-contained component. We know this somehow offers a lower level of integration and less freedom, but also ensures a more immediate approach (just add the basic component to the webpage and it is ready-to-go) and a higher reusability (thanks to being self-contained).

The 3DHOP scene is composed of different elements: the *mesh* and the *instance* are the most basic. A *mesh* is simply a 3D model (single or multi-resolution). An *instance* is an occurrence of the mesh in the scene. This separation seems an unnecessary complication, given that the tool aims to be simple, but it is nevertheless the simplest way pursuing an efficient geometry instancing, especially when multiple objects share the same geometry.

Meshes and *instances* may have an attached transformation, that may be specified in three different ways: setting the translation, rotation and scale components (through 3-number vectors which determines the translation on the axis, the Euler angles of rotation, or the scale factors along the three spatial directions); as a roto-translation matrix (a 16-number vector); or by using the predefined SpiderGL functions. The most obvious use is to exploit the mesh transformation to bring the 3D model into a basic position/orientation (e.g. to put a 3D model originally not perfectly aligned to its axis into a "straight" position) and then to locate each *instance*, to set its specific position/orientation/scale.

An example of declaration of *meshes* and *instances* is the following:

```
meshes: {
  "Laurana": {
    url: "singleres/laurana.ply" },
  "Gargoyle": {
    url: "multires/gargo.nxs" },
```

```
  "Box": {
    url: "singleres/cube.ply",
    transform: {
      matrix:
        SglMat4.scaling([13.0, 0.5, 10.0])
    }
 }
},
modelInstances: {
  "Lady": {
    mesh: "Laurana",
    transform: {
      matrix: [1.0,   0.0,  0.0, 0.0,
               0.0,   1.0,  0.0, 0.0,
               0.0,   0.0,  1.0, 0.0,
               0.0, 235.0,-50.0, 1.0]
      }
    },
  "GargoRight": {
    mesh: "Gargoyle",
    transform: {
      matrix:
        SglMat4.mul(
          SglMat4.translation(
            [120.0, 0.0, 150.0]),
          SglMat4.rotationAngleAxis(
            sglDegToRad(-90.0),
            [0.0, 1.0, 0.0]))
    }
  },
  "GargoLeft": {
    mesh: "Gargoyle",
    transform: {
      translation : [-120.0, 0.0, 120.0]
    }
  },
  "Base": {
    mesh: "Box",
    transform: {
      matrix:
        SglMat4.translation(
          [0.0, -12.5, 0.0])
    }
  }
},
```

In this example a few simple elements are instantiated and arrayed in space, with the corresponding scene visible in Figure 3.4. A *mesh* element having the shape of a cube is firstly scaled to become the base of the example 3D scene, and then positioned at the *instance* level. The other models are arranged (translated or rotated and translated) onto the base at *instance* level; the two gargoyles share the

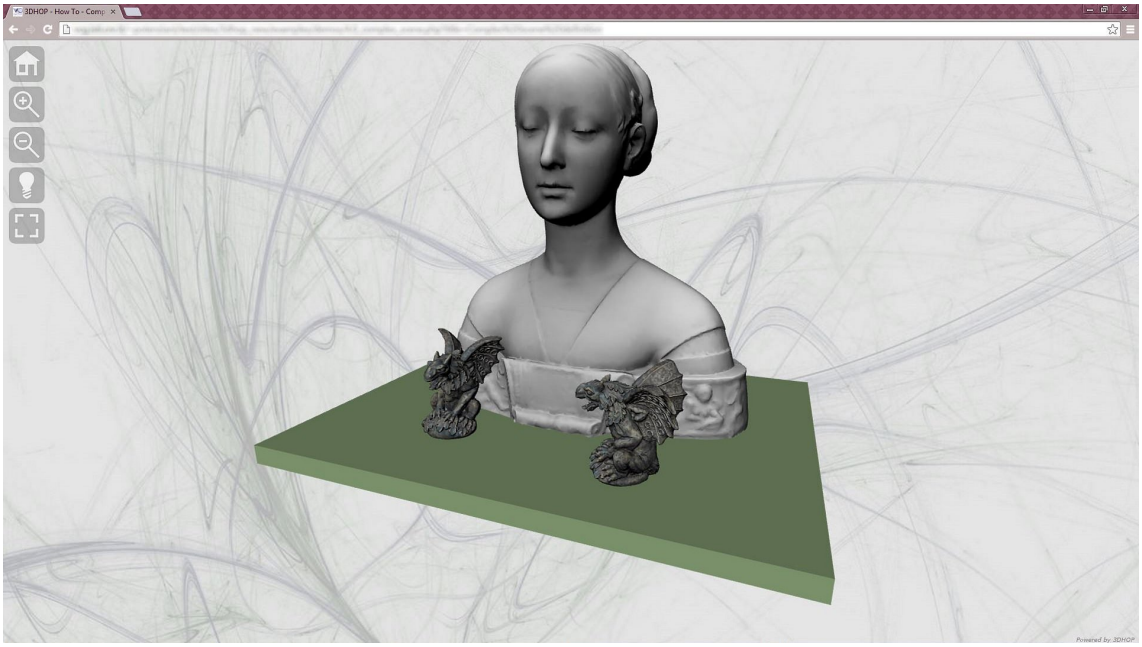

Figure 3.4: A simple scene in 3DHOP [Vis14a] created by instancing geometries and applying transformations.

same *mesh* geometry.

A 3DHOP scene includes many other elements, which are presented in the following: e.g. the *trackball* (used to drive the interaction) or the *hot-spot* elements used for picking. General scene parameters (e.g. the field of view and the custom scene centering) are also declared in the same way.

The declarative approach also has the advantage of more easily managing content retrieved from a database. The scene description is a JavaScript structure which can be easily filled with data retrieved by a query to a database; this would be less straightforward using an imperative-like setup.

3.3.3 Interaction Components

A 3D viewer is not just a rendering engine, but also includes the components required to implement the user interaction. 3DHOP mostly uses the *World In Hand* metaphor (see §2.4.3), where the camera is fixed and the object is manipulated by the user in front of it, generally using a trackball.

It is difficult, if not impossible, to create a single all-purpose trackball, able to cope with the specific geometric characteristics of every possible object. For this reason, we decided to implement a series of basic trackballs, letting the user to choose the more appropriate one. At the moment, the 3DHOP distribution includes four different trackballs (others could be added in the future):

- *Full-Sphere*: it is the trackball providing the more free interaction. It enables

the user to rotate the object around all axes at the same time.

- *TurnTable*: this is the most accessible trackball, especially for final users not used to interacting with 3D scene. It provides rotation around the vertical axis and tilting around the horizontal axis. With this trackball it is possible to reach almost all view positions around an object in a simple way, maintaining its verticality (e.g. preventing to rotate a statue head-down, feet-up).

- *TurnTable-Pan*: this trackball is the most flexible one, and it is an extension of the previous one. It add the feature that allow the user to pan across the object, changing the center of rotation of the scene. It gives more freedom to explore the objects, especially when zooming close to the surface.

- *Pan-Tilt*: this trackball is the more specialized one. It is tailored to present bas-reliefs or objects whose detail is mostly located on a single plane.

Having a series of basic trackballs, implemented with simple, open and documented code, will allow developers to add new interaction modes coping with specific visualization needs. For this reason, each trackball in the distribution is a separate file, making it easier to use them as a codebase.

Trackballs can be configured with limits on their axes, to restrict the position reachable by the user. This is useful to avoid the user going, for example, below ground level in buildings, or behind objects with only a frontal part (like bas-reliefs). Trackballs can be also animated (we present an example in §3.3.4).

In each 3DHOP viewer/installation there is only one trackball selected (*TurnTable* trackball is the default). To explicitly choose and configure a trackball, the developer has to specify the *trackball* element of the scene:

```
trackball: {
  type: TurnTableTrackball,
  trackOptions: {
    startPhi      : 0.0,
    startTheta    : 0.0,
    startDistance : 2.5,
    minMaxPhi     : [-90, 120],
    minMaxTheta   : [-10.0, 75.0],
    minMaxDist    : [0.5, 3.0]
  }
}
```

In the example above, the developer has chosen a *TurnTable*, starting exactly in front of the object (*phi* is rotation around vertical axis, *theta* the elevation angle) but a bit far from the object (distance 2.5 means that the camera distance is 2.5 times the size of the object bounding box). The trackball is limited both in the horizontal rotation (a bit to left, more to the right) and in the vertical one (not much below, a lot above); it is also impossible to go nearer than 0.5 and farther than 3.0 units from the object (again, expressed in multiples of the object size).

Like in all configurations of 3DHOP components, it is not needed to specify *all* the parameters, since the unspecified ones will retain their default; it is sufficient to specify only the ones that need to be changed.

This approach, based on the trackball metaphor, is perfect to manipulate "objects", but it makes it much more difficult to navigate other types of geometries (such as buildings and terrains). "Ad-hoc" interaction components more suited for exploring these complex scenes (such as maps-like approach for terrain models, or waypoint-based path for the interior of a building) are currently in testing phase.

3.3.4 DOM Integration

As introduced before, we wanted to create a framework offering basic viewers (if no other functions are configured), but also visualization components able to interact with the rest of the webpage. To this aim, we added a series of exposed functions and events, usable by a developer to allow 3DHOP components to interact with the rest of the Web page logic. Our idea was to implement multiple, self-contained functions, with no high-level semantics attached, in order to provide the developers with a toolbox.

Trackball Automation

The most basic interaction between a Web page and the 3D visualization component is the control of the trackball. 3DHOP trackballs are able to give feedback on their current position: an exposed JavaScript function (*getTrackballPosition*) returns a structure containing the current state of the trackball. Another provided JavaScript function (*setTrackballPosition*) can be used to instantly move the trackball to a specific position by feeding it with a new state description. Additionally, it is possible to *animate* the trackballs to reach a certain position: instead of instantly changing its state, the camera follows a smooth animation path linking the current position with the specified one. These functions allow the content creator to build, for example, a bookmarking mechanism for pre-selected views, a "share this view" button or a guided animated tour around the object. An example is shown in the *Helm* viewer (Figure 3.5), where the buttons on the right side of the window move the trackball to the views visually represented by the small icons.

Visibility Control

Most visual presentation tools implement the control of the visibility of the different models. Model instances in 3DHOP can be configured in order to be visible or invisible at startup (visible is the default), and their visibility status can be changed at runtime using specific JavaScript functions exposed by the tool. The visibility feature can work both on a single instance (using the instance name as selection ID), and on a group of these (using a tag-based system as selection ID).

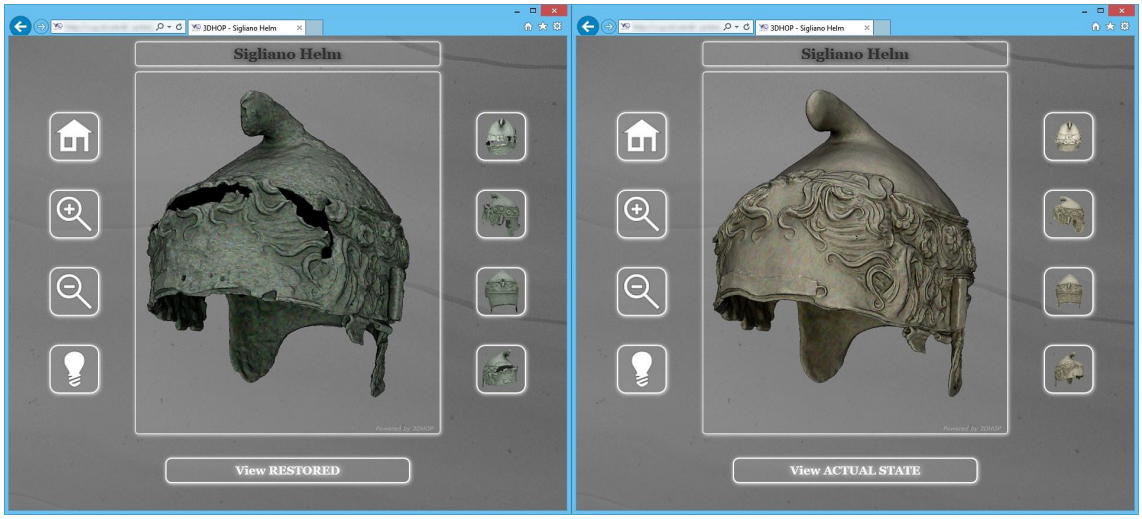

Figure 3.5: The *Helm* viewer: 3DHOP [Vis14a] allows to inspect an Etruscan helm either in its current state (image on the left) or in its virtual restoration version (image on the right). The user may switch between the two versions (using the ViewRestored/ViewActualState button), explore the model (it adopts the TurnTable trackball), and use the links on the right side of the window to go to interesting views of the model (these buttons will animate the trackball to reach the selected view position).

The tag-based selection is an interesting trick that allows the visibility functions to act on all instances that have a specific tag. Model instances have a *tags* property, which is basically a series of strings. We can assign to each instance the tag of each "group" it belongs to or, if necessary, a unique tag. Using this simple mechanism, it is possible to address single entities as well as groups.

3DHOP exposes a function to set visibility and another one to toggle the visibility of a set of instances. For example, the *Luni Statues* viewer (Figure 3.2) presents four statues, each one composed of an original part and an integration; it is possible to make visible/invisible each statue either as a whole, or all the original parts or all the integrations of the entire set or, finally, the original/integration parts of a specific statue. The original part of statue #1 has tags [*"figure1", "original"*]; the integration part of statue #1 has tags [*"figure1", "integration"*], and so on for the other figures. Therefore, starting from a 3D scene where all the models are visible, in order to make visible only the whole statue #1, the developer will use these calls:

```
setInstanceVisibility(HOP_ALL, false, false);
setInstanceVisibility("figure1", true, true);
```

Conversely, to show only original parts for statue #1 and #3:

```
toggleInstanceVisibility("integration", false);
setInstanceVisibility("figure2", false, false);
setInstanceVisibility("figure4", false, true);
```

where HOP_ALL is a constant used to select all of the instances; the second

parameter of *setInstanceVisibility* is the new visibility state; and the last parameter of both *setInstanceVisibility* and *toggleInstanceVisibility* functions is used to force a redraw.

Of course, the tag selection system does not prevent using instances name IDs to drive the visibility status of a single 3D object. Also in this case 3DHOP exposes a function to set and another one to toggle the visibility of the instances (these methods work in the same way of the tag-based ones, with the only difference to accept as input the instance name string instead of the tag string). "By name" visibility controls are used in the *Helm* viewer (Figure 3.5) to switch between the helm before and after restoration: in this example there are two *instances* of different *meshes* in the same positions, and to switch between the two, one is hidden while the other is shown.

Hot-spots and Picking

Another widely available feature in webpages is the presence of clickable *hot-spots*. This feature is often connected to something happening in the 3D visualization or elsewhere in the webpage. Depending on the visualization scheme, it may be interesting to have a picking component able to detect a pick on a hot-spot, but also to detect a pick on an instance of a 3D model. 3DHOP does support both levels of interaction. In order to use this feature, the developer shall use two JavaScript functions to handle the picking (of hot-spots and instances) and register these two functions to the handles exposed by 3DHOP. The first function (hooked to *onPickedInstance*) is invoked every time a model instance has been clicked, and returns the name of the picked instance. The second one (hooked to *onPickedSpot*) is invoked every time a hot-spot is clicked, again returning its name. A third function, which returns the exact XYZ coordinate of the clicked point has been also developed, and it is currently available in all the latest 3DHOP releases.

In order to be more flexible, instead of just a single point, a hot-spot may have an arbitrary shape and geometry. This is obtained by associating a *mesh* to the hot-spot, similarly to the way a 3D model is specified when declaring an instance (a geometry is declared as a *mesh*, and then used in the declaration of the hot-spot). In the simpler cases, a hot-spot can be defined using a sphere or a cube model, moved to the correct position and appropriately scaled. In more complex situations, the user can provide a specific geometry, for example created using a 3D modeling tool. Picking is implemented using a basic CG method: when picking, the scene is rendered in an off-screen buffer, with each pickable object rendered as a solid unique color, which encodes its ID, while non-pickable objects are rendered solid black. The picked pixel is retrieved from this buffer: if black, nothing has been picked; if non-black, the color is transformed back into the ID of the picked object. This method does not require too many resources, and works pretty well also on complex scenes. The picking mechanism also works in real time when the user moves the mouse, thus obtaining an "onOver" hook, and enables the hot-spot geometry to light up.

This feature may be deactivated when the scene is too complex, to speed up the rendering.

Hot-spots may be made active or inactive using a tag-based mechanism similar to the one used in the visibility control, making it possible to define "hot-spot groups" which can be independently activated/deactivated (e.g. to show different layers of information or linking). Each hot-spot may have a specific color and an associated cursor.

An example of this kind of interaction is provided in the *Capsella Samagher* viewer (Figure 3.6): in this example, when a hot-spot is picked some related information (an image and a descriptive text) is shown in the left-most portion of the Web page, and the view over the 3D model is moved to better frame the detail (using the trackball animation feature).

Figure 3.6: The *Capsella Samagher* viewer: in this 3DHOP [Vis14a] example, the antique reliquary and its lid are presented with hot-spots (light-blue regions). The hot-spots, when picked, centers the view over the hot-spot area and show the corresponding descriptive content (images and text) in the left-most part of the webpage.

3.3.5 Publishing with 3DHOP

The trade-off between ease of use and flexibility is a major issue when creating a tool for non-expert developers. If the features are too simple or restricted, users with particular needs may not find proper support; on the other hand, an increase in flexibility could reduce simplicity of use. For this reason, the 3DHOP tool has been designed with different levels of access, to be as straightforward as possible for the

more simple cases but, at the same time, able to provide enough configurable features to support the huge variability of Cultural Heritage artworks and applications. Users with knowledge of JavaScript programming and Web design will have no problem in using the framework, since its basic paradigm mimics the one normally employed in standard Web development.

3DHOP for Unskilled Content Creators

Developers with limited programming skills may still use the framework using one of the following strategies:

- **Zero configuration**: since all the components have a set of safe defaults, it is possible to create a visualization page without configuring anything. This "minimal" visualization page is contained in a folder of the distribution, and can be readily used by the most inexperienced of users, since it is only necessary to change the 3D model file.

- **How-To**: in addition to plain documentation, we opted to present the different features in a specific *How-To* section that provides descriptions of the visualization component detailing the parameter-based configuration. These pages contain reusable examples that can be modified following the content of the how-to.

- **Templates**: in the *Gallery* page of the 3DHOP website, it is possible to find various examples (with different levels of complexity) which cover typical cases of usage in the CH field. The idea is to provide the developers with non-trivial usable templates, which can be used or customized with just minimal changes. After changing just the 3D model file (and the graphic elements, if needed), a completely unskilled developer may create their own visualization page without even modifying the HTML code. We are currently working on better documentation for the templates, and on cleaning-up their HTML code for simpler use.

3DHOP as a Codebase

Though the 3DHOP tool is able to provide even unskilled content creators with a basic and universal high resolution Web3D viewer, there are many projects where more specific solutions are needed to fully exploit the data and to reach the communication goals. In these cases, 3DHOP may also be seen as a codebase for the creations of more skilled users.

In fact, the modular structure of the tool facilitates the implementation of new, specialized components, or the tuning of existing ones, giving to CG programmers and/or Web developers the chance to heavily modify 3DHOP to cope with the particular needs of a project. The modules constituting 3DHOP have been thought

to be organized in layers, which starting from the HTML high level definition of the viewer (top layer), trough several steps arrive until the lower level JavaScript libraries (innermost layer) responsible for the viewer basic behaviors.

More in detail, 3DHOP relies on two main modules (that can be imagined just underneath the HTML layer): the first one is related to the 3D component of the viewer (scene setup, rendering, streaming, etc.), and the latter responsible for the its interface. These two modules (called "presenter" and "init", and corresponding to two specific JavaScript files in the 3DHOP distribution), represent the firsts elements to edit for a skilled creator looking for a deep 3DHOP customization.

The "init" interface module is essentially an independent element (3DHOP could work properly even without any interface), and represent just one of the infinite possibilities for building an interface on the top of 3DHOP. Strongly related to the CSS definition of the viewer, and with the skin elements in its distribution, it mostly depends on the jQuery library [Res06] (lower layer of the viewer structure) for activate all the appearance effects characterizing the 3DHOP interface.

Conversely the "presenter" module represents a core element of the viewer, mandatory in any kind of visualization. Indeed, it contains the parser for the 3D scene setup, as well as the definition of the shaders responsible for the 3D scene appearance. It uses the SpiderGL library [Vis10] (lower layer of the viewer structure) for accessing the WebGL API [Khr09], and is linked to a number of 3DHOP sub-modules devoted to drive specific viewer peculiarities.

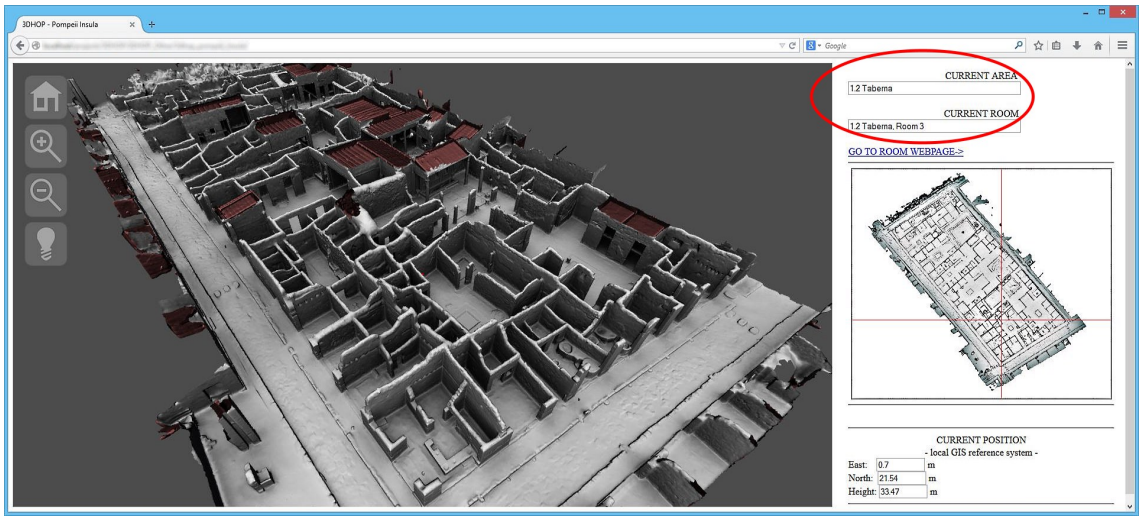

Figure 3.7: The *Pompeii* explorer: in this experiment 3DHOP [Vis14a] allows to explore the entire Insula V 1 of Pompeii. Navigation is controlled by mouse inputs (using a custom terrain-enabled trackball) or by clicking on the 2D MiniMap (see on the right of the window). The viewer keeps track of the current location of the user, showing the name of the room and of the house (text fields circled in red in the image).

An example of sub-modules are the interaction elements. Defined in 3DHOP

as trackballs (see §3.3.3), these modules specify different interactions with the 3D scene. At each trackball corresponds a single JavaScript file, to make easier for the content creator to add, remove, or modify each of them. Another example of sub-modules are the model importers. 3DHOP provides two modules aimed at handling the supported formats: PLY for single-resolution models and Nexus for the multi-resolution ones (see §3.3.1). Also in this case the two modules are specified in two different JavaScript files to make easier modify or expand the original format loaders configuration.

Modifying the modules composing the layered structure of 3DHOP, starting from top and eventually going down, is the best way in using 3DHOP as a codebase. An example of this modification strategy is the application designed for the Web-based exploration of an entire *insula* (an area surrounded by four major streets) of the Pompeii archaeological site. In this case the basic version of 3DHOP was used as a starting point to create a customized viewer for the *Pompeii* model, presented in Figure 3.7.

The added value of this specific modification is the work done to extend the basic trackball to an interaction interface suited to the exploration of terrain-with-structures models. This system offers a double interaction method: a *bird-view* navigation and a *first-person-view* navigation. Both navigation methods are able to follow the height of the ground level, and collision detection with walls is available in first-person navigation. This new 3DHOP incarnation features also a new component: the interactive *MiniMap* (an HTML5 canvas entity, already introduced in 2.4.3) placed on the right-most portion of Figure 3.7. In each instant of the navigation, the current position of the viewer is shown on the map; clicking on any location in the MiniMap, the viewer is virtually moved to the desired location. Moreover, the system keeps track of the position of the viewer, not just showing the user location on the MiniMap, but also showing the name of the specific building/room the user is currently visiting (see the two textual fields on top-right, circled in red in Figure 3.7), retrieved from an existing Web repository.

3.4 Results

3DHOP is a solution that aims at providing an easy way to create advanced 3D Web publications, offering accessibility at different utilization levels. Its structure has been designed for being at the same time a perfect enabling solution (for the Web3D community) to fill the 3D Web publishing gaps outlined at the beginning of this Chapter, as well a helpful tool (for the CH community) to create and share advanced content on the Web, usable not only for dissemination purposes, but even in experts and practitioners work-flows.

3DHOP is an ongoing project: it has significantly grown since the first release, and still is going to grow. Since it is a modular framework, there are many new components (or variations of the existing ones) that can be added to support the

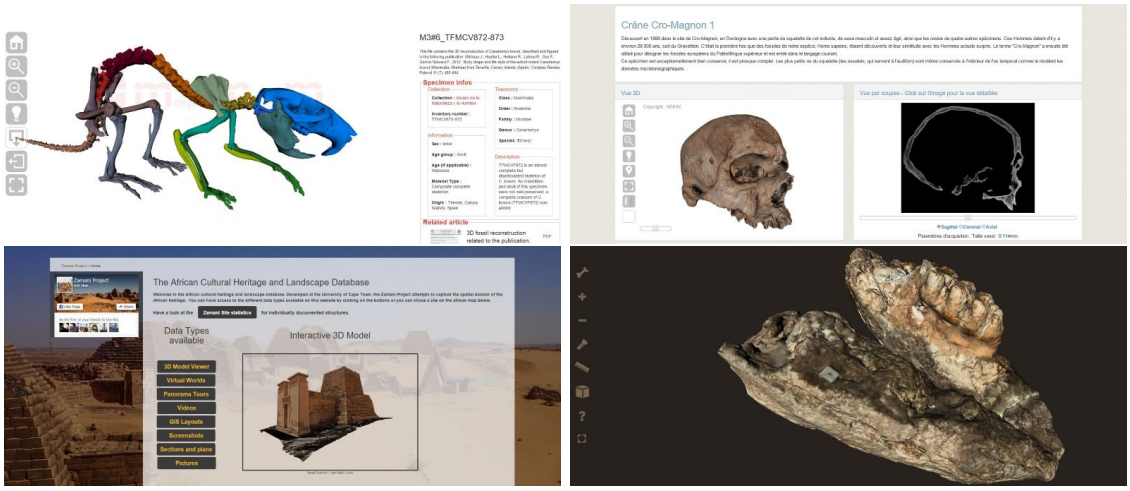

Figure 3.8: Four examples of independent projects developed by the community using 3DHOP (in clockwise order starting from the upper left): the *MorphoMuseum* [LO14, LO16] platform: publishing and sharing online 3D models of vertebrates (the panel on the right contains specimen infos and links to the related article); the *Cabinet De Curiosites 3D* [Mus15] repository: presenting a virtual exhibition of 3D models, each one paired with several informative media (text, images, etc.); the *Megafauna 3D* [Uni16] experience: discovering the 3D fossils of the animals that lived in South America 10.000 years ago (the interactive platform is mostly aimed at educational purposes); the *Zamani* [Uni14] project: providing metrically accurate digital representations of African historical sites on the Web (this example seeks to provide data for research and education, for restoration and conservation, increasing awareness of tangible Cultural Heritage).

creation of more flexible and effective interactive visualizations.

Some of the 3DHOP future enhancements and extensions could refer to these features:

- *Novel navigation and visualization components*: new trackball types and new scene manipulation functions are on the development list. Examples are the trackball used in the *Pompeii* explorer (Figure 3.7), supporting fly-over and walk-on-the-ground modes, that will be documented and added soon to the *Gallery* and the 3DHOP official distribution. Moreover, all geometries are currently rendered using a few basic shaders. Our goal in the near future is to provide different, configurable shaders, which should be selectively attached to each instance.

- *Dynamic definition of scenes*: at the moment, the scene definition is *static*. Once declared in the initialization, there is no direct way to modify the parameters of the different entities (even if, the modular framework structure, actually allow to indirectly act some dynamic changes). Our

development road-map aims at reaching this functionality in a progressive way, starting from being able to modify the associated transformations, then to move to the other properties, and ending with the ability to dynamically add/remove entities.

- *Additional types of datasets*: in the context of Web visualization, other types of datasets could be effectively integrated into 3DHOP. An example is represented by the handling of terrains as *Digital Elevation Models* (DEM). DEM are defined in a 2D 1/2 space and can be managed more effectively than a 3D model using specialized strategies. Exploring the possibility to handle a Web-streamable multi-resolution representation (based on quad-trees) of a terrain geometry could be very interesting for 3DHOP. Indeed, adding a DEM to a 3D scene may be useful to better cope with applications that involve landscapes of archaeological interest.

- *New specialized analysis tools*: currently 3DHOP provides several analytical components specifically designed for the CH domain: directional light control, measurement tools, planes sectioning, etc. In the future could be valuable to integrate in 3DHOP other similar tools enabling more specialized and complex analysis. A representative example of that may be represented by a comparison tool, able to simultaneously and synchronously visualize two different 3D models (superimposed or side to side), possibly also providing visual effects for highlighting the differences between them, so enhancing the possibilities to visually compare similar instances of the same object.

We conceived 3DHOP as an open source tool (the source code is available online at [Vis14a]), so the extension and modification of the framework has been highly encouraged since from the firsts implementations. Indeed, we believe that 3DHOP has the potential to sprout an independent community of users, that could share examples, exchange experiences, and create connections.

Following the first release we have been contacted by several users willing to test and evaluate the framework, thus we are gathering suggestions and feedback. Some of the most relevant third parties implementations of 3DHOP are reported in Figure 3.8.

These external exploitations, together with some independent certificate of merit (3DHOP got the honorable mention in the category "Best DH Tool or Suite of Tools" of the *Digital Humanities Awards 2015*), seems to suggest a positive appreciation of the 3DHOP design choices.

The results of this experience of design, development and testing, has been published here:

- *3DHOP: 3D Heritage Online Presenter*, Potenziani M., Callieri M., Dellepiane M., Corsini M., Ponchio F. and Scopigno R., International Journal Computers

& Graphics, Elsevier, ISSN: 0097-8493, Volume 52, page 129-141, November 2015; [1]

The 3DHOP sources, a demonstrative gallery of examples, a complete set of how-tos descriptions, the API docs, and additional contents, can be accessed at the project official website [2].

[1] © 2018 Elsevier. Text and figures reproduced by kind permission of Elsevier.
[2] `http://3dhop.net`

Chapter 4

Evaluating 3DHOP as a Specialized, Direct Publishing Tool

We have designed the 3D Heritage Online Presenter as a tool able to extend the possibilities of efficiently creating specialized 3D content on the Web; and a core point in this design was to make it a flexible tool, able to cope with simple publishing tasks as well as complex, customizable solutions.

As we have already seen in Chapter 3, despite being a fairly new tool, various examples of visualization pages built using 3DHOP can be already found online. The fact that the majority of this pages have been developed by independent users and content creators (i.e. without our help or support), can already be seen as supporting our design choices, in terms of accessibility and ease of use. However, most of these publishing efforts are limited to the creation of basic applications, in which the viewer is used just a "dumb" container for the 3D content, with a limited customization and poorly specialized on the presentation aims.

Since our aim was to design a tool that could be flexible enough to be heavily customized for more specialized interactive presentation experiences, we think that, to prove this point and the actual consistency of the design choices taken when building our framework, a more comprehensive and thorough evaluation of the 3DHOP capabilities is needed. For this reason, we have contributed to the implementation of several publishing experiences aimed to test the real effectiveness of the 3DHOP features previously introduced. Among those projects, we present here two relevant testbeds, each focused to address different communication needs.

Both the proposed case-studies explore the possibilities of a specialized Web3D content creation, but while the first one propose an example of publishing aimed at supporting a museum exhibition (dissemination to the public), the second one has been developed for supporting scientific analysis (we are still limiting our test to the *direct* publishing, i.e. the act of creating a visualization for a specific dataset, while

we will explore the concepts of assisted, service-based and bulk publishing in the next Chapter).

More specifically, the first contribute we present retraces the steps of a publishing campaign that, starting from a very peculiar artifact (a modern-art painting), 3D digitized for conservative and diagnostic purposes, will lead to the design of a Web3D application to use in a virtual exhibition, as well as museum kiosk in a real exhibition. In this case study, the 3DHOP code has been quite customized, obtaining a framework tailored on the most interesting aspects of the analyzed artwork. The resulting application tries to exploit the expressive power of the 3D geometry presenting these information from an unusual point of view, with the final goal to improve the visitors understanding and awareness about the painting and the process of its creation.

In the second contribute, instead, we present a publishing experience aimed to a more technical data presentation. In this case the main goal was to build a tool supporting experts in their work of study and documentation of an archaeological find. Starting from the availability of the digitized 3D geometry of the artifact, together with a wide and heterogeneous set of analytical examinations on it, we designed, through a substantial customization of the 3DHOP user interface, a specialized Web3D application able to effectively present all these information. In the resulting Web viewer the 3D models is used as main channel to manage and display all the achieved scientific analysis and historical information about the artifacts, integrated in a single reference system.

The introduced publishing actions, even thought aimed to cover different needs, are characterized by a common "fil rouge". These case studies have both been built around the (high-resolution) 3D models, paying a particular attention (as we will see in the following) to all the process leading to the generation of the 3D geometries. This data-centric design has allowed us to exploits and evaluate the modularity and the flexibility of the 3DHOP solution, cutting, pasting, and modifying the framework features to fit the specific aims of the two experiences.

The positive assessment, in real-world applicative environments, of the outcomes of this design approach, seems to validate the effectiveness of the choice endorsed developing the proposed solution.

4.1 Alchemy in 3D: A Digitization for a Web-based Journey

The primary purpose of a specialized Web3D publication very often is related to dissemination. Thus, in this particular case study [Vis15c, CPP+15], we will propose a real publishing action that, by exploiting the digitized 3D geometry of an important modern-art painting (Alchemy, by Jackson Pollock), will present to the museum public, with an engaging and immersive point of view, all the hidden aspects of this

painting.

This journey starts with the accurate digitization and the careful processing, aimed to generate a very precise high-resolution 3D model that proved to be useful in different stages of the diagnostic and conservation campaign (the 3D model was integrated in the conservation process, along with the other diagnostic investigations), but not only in this step. Indeed, the most interesting use of the 3D model (at least for the purposes of this discussion) was in disseminating the conservation and the diagnostic campaign to the museum visitors. This was done following the idea of going beyond photo-realism and the use of the scanner-measured geometry to try to interpret and understand the traces and signs on the surface of the painting, in relation with the gestures and techniques used by Pollock while painting this masterpiece. Combining the knowledge of the curators and the metric data gathered in the digitization, we were able to discover and validate several interesting aspects of the painting, in the direction of trying to better understand (and present) the painting process which was, in the idea of the artist, an essential part of the artwork.

3DHOP was selected to correctly convey this vision. It was used to create an interactive kiosk and a Web application, to have the visitors navigate the 3D model and interact with the artwork accessing the information related to the most relevant geometrical details.

4.1.1 Overview

3D digitization technologies have found a fertile ground in the Cultural Heritage field, and have proved to be useful for technical applications in the study, conservation and restoration of artifacts. 3D digital models are also used a lot in the presentation of artworks, especially when targeting a wide audience.

In the former case, for the technical use, 3D is regarded mostly as a source of "pure data", in terms of measurements, technical data and metric geometrical information. In the latter, the visualization and presentation exploits the "visual" side of 3D models, aiming at reaching the highest possible level of realism, in order to give the user the impression of looking at the original object. The important fact (which makes 3D so interesting) is that if the 3D data is captured properly, the very same information may be exploited in these two ways.

Sometimes, these two opposite facets of the 3D data can be exploited together leading to interesting results. In this specific project, for example, we tried to use the *visual geometrical* facet of the 3D digital model to better *understand* and *present* the painting Alchemy by Jackson Pollock (Figure 4.1).

Our idea was that, by isolating the geometrical component of the painting, and looking at it with the trained eye of the 3D expert and of the curator, it would be easier to read and interpret the signs and traces of the painting process, fulfilling the idea of the artist, that the painting process is itself part of the artwork. Once done that, we used those informations to design and build a 3D presentation (on-site and

Web) tailored on the painting geometry, so as to explain to final user the gestures, actions and techniques belonging to Jackson Pollock at work.

Alchemy Under Scrutiny

Alchemy by Jackson Pollock (Peggy Guggenheim Collection), is one of his most famous works, and a major contribution to twentieth century art. Dating back to 1947, it was one of its first experiment of what will be later called *"action painting"* and *"dripping technique"*. For the first time, Pollock placed the canvas on the ground, fixed on a wooden frame, and proceeded in pouring/dripping/squeezing paint on its surface.

Alchemy, being one of its earliest experiments, presents a noteworthy variety of painting materials (metallic paint, industrial enamel, acrylics), of non-paint media integrated in the structure of the painting (sand, pebbles, fibers, wooden sticks); all these elements were laid out using a plethora of different application methods (spatula, brushes, squeezed from the tube, dripped, splashed out of a syringe).

The traditional procedures of painting were revolutionized in the sense that the action of laying out the paint became itself an artwork, while the final appearance of the painting was not related to the notion of "descriptive shape" anymore, but was basically a record of the act of painting. Similarly to a musical record, the traces of colors, paints and materials on the surface should somehow *play back* the gestures and movements of the artist at work. For this reason, looking at Alchemy just as a "flat painted image" cannot fulfill the artistic aim of the painting, making impossible for the visitors to get in touch with the vision of the artist.

Figure 4.1: Alchemy, by Jackson Pollock.

Our work took place in the framework of a larger project of conservation and

diagnostic investigations. While being a relatively recent work of art, Alchemy was in need of an accurate monitoring and conservation. The exposition of the painting without protection in the gallery had caused an accumulation of dust and an opaque patina was coating the paint, requiring a deep cleaning. The painting has a huge amount of paint on the canvas, more than 3kg, that exerts on the canvas and on the wooden frame a much stronger pulling force than a "standard" painting; the canvas and the wooden frame had to be checked for signs of stress. Finally, Pollock has used for this painting unconventional paints, made for the automotive industry, or for indoor use and decoration, and certainly not formulated to be mixed together. Their duration in time and their chemical interaction is unclear, and their state of conservation had to be checked.

These reasons made the conservator of the Guggenheim collection decide to carry out a conservation and diagnostic campaign, with the goal of better understanding the artwork, while at the same time cleaning it. One of the diagnostic investigations carried out in the campaign was the 3D digitization of the painting. The high resolution 3D model that was created was used in different moments of the conservation campaign, and it played a main role in the final exhibition that was prepared to present the results to the public.

4.1.2 3D Data Acquisition and Processing

While the digitization and 3D model is not the primary focus of this experience, it is anyway necessary to spend a few words on the acquisition and processing to better understand the extent of the employed 3D model. To fulfill the designed roles of scientific/geometric data to be interpreted by experts, and of visually rich representation to be exposed to the visitors, it was imperative to have a high-quality data.

Alchemy exhibits a surprisingly detailed geometry, and the presence of paints with a quite diverse optical behavior (metallic, shiny, translucent) posed severe difficulties for the digitization. Considering our target resolution, the size of the painting was also non-trivial (221x114 cm). The acquisition took place *before* the cleaning. This timing proved to be an advantage: the presence of dust and of the opaque patina, while not changing the underlying geometry, helped a lot in overcoming the problems of specular reflections and of metallic paint scattering.

We chose to use a structured light scanner, this technology could give us the required level of detail (0.1 - 0.2 mm) and is a bit more resilient to specular reflections with respect to other acquisition techniques. We used a GOM Atos scanner (see Figure 4.2). Each single shot covers an area of around 60x40 cm at a resolution of 0.2 mm (25 points per mm^2). In order to cover the whole surface of the painting, including its sides, with enough overlap to ensure rigidity during alignment and data redundancy to reduce sampling noise, we took 72 scans, for a total of nearly 120 millions of points. Two areas were acquired at an even higher resolution (100 points per mm^2).

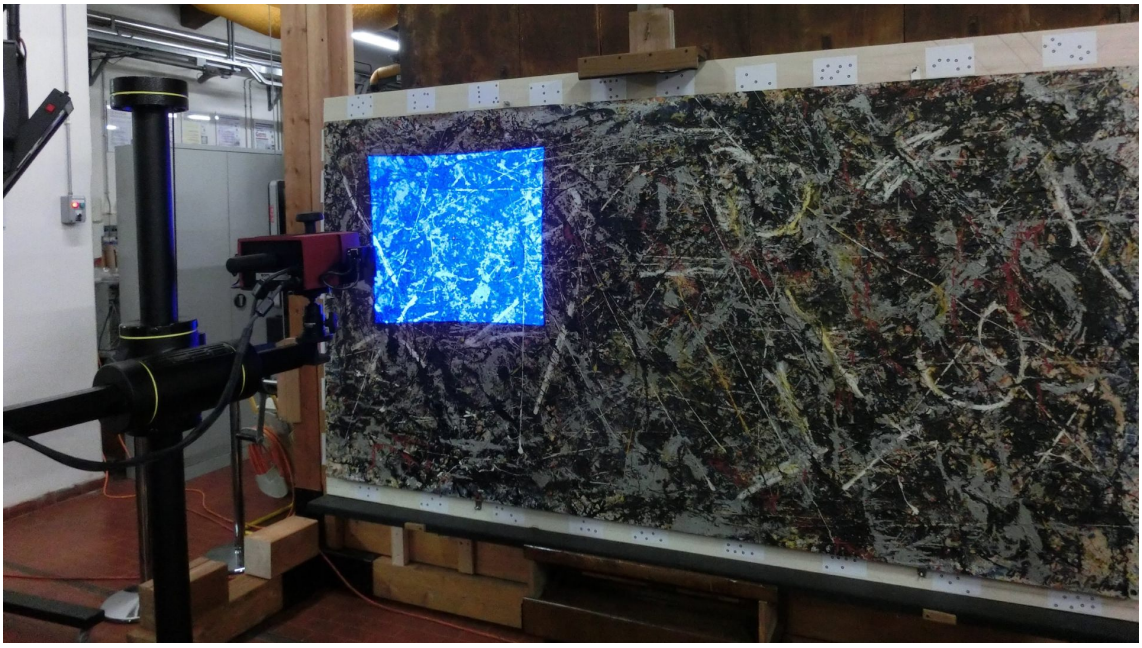

Figure 4.2: The 3D scanning of the painting.

The raw 3D data was processed with the MeshLab [Vis05] software, used for data cleaning, alignment of range maps, creation of the triangle mesh and color mapping. The result of the processing is a 3D model composed by 80 millions of triangles covering the whole painting, plus two models at higher resolution for the areas of detail, of around 100 millions of triangles each. On top of the obtained geometry, we mapped a high resolution image (50 MPixels) obtained with a multi-spectral scanner provided by the Istituto Nazionale di Ottica (INO-CNR), thus encoding in the digital model both high resolution geometric and color information (see Figure 4.3).

4.1.3 Which Use of the 3D Model?

As said before, the main and general uses of 3D models are related to its technical and scientific data, and to its ability to create photorealistic visualization. In this sense, this project was not different.

The obtained 3D model was part of the diagnostic data gathered during the campaign. It has been used, during the cleaning and the other analysis to help guiding the cleaning process, to validate hypotheses and to map the local information acquired with other devices. The 3D model of an object has a strong potential as a "reference system", as it is possible to map other types of information (for example, the local analysis of the materials) onto the surface. As usual in these cases, the 3D data produced (raw scans and 3D models) will also be stored in the museum archives, to be used as a technical documentation for any following conservation and restoration actions, as it represents a precise, scientific snapshot of the state of the

Figure 4.3: A rendering of the final 3D model with a shifting material, showing (from left to right) both pure geometry and color.

artwork.

At the same time, the 3D model has also been used to create realistic visual representation of the artwork. This included high-resolution images and photo-realistic video sequences, to be used as a dissemination material for the museum audience. These visual presentation techniques were used in the creation of the video used in the exhibition and in the dissemination website (see §4.1.4).

These uses were indeed valuable for the campaign and the dissemination, but are pretty standard in modern digitization projects; thus, due to the peculiar nature of the artwork, we decided to use the same data also for a different kind of "understanding" of the painting.

Understanding the Painting Through the 3D

A visual inspection of the geometry of 3D digitized models is quite common, but is generally focused towards more "metric" considerations. In this case, we argue that the close inspection of the naked geometry of Alchemy is the best way to study and understand the painting. Although looking at the naked geometry of a painting to understand it may seem against reason, it has to be considered that, while the color was chosen and used by Pollock for specific reasons, it is also true that only the shape of the paint on the canvas is able to effectively describe the gesture and the act of painting, so important for this artwork.

Indeed, considering this painting as a flat image is misleading, as the final

colored appearance does not convey the full story. Conversely, looking at the naked geometry, brings out much more clearly the underlying structure of the traces of paint. For this reason we also employed a lot the non-photorealistic rendering techniques to make the geometry even more readable, taking further distance from the photo-realism paradigm.

The 3D models (the whole artwork at 0.2 mm and the detail areas at 0.1 mm) had more than enough detail to isolate the traces of paints and their superimposition order, so helping to understand the motions and the actions of Jackson Pollock, as visible in Figure 4.4.

Figure 4.4: A close-up of a 7x15 cm area of the painting. Even this small sample contains examples of paint squeezed out directly from the tube, oozed from a syringe, dripped from a brush or stick, and the presence of small pebbles.

During the cleaning and the design of the exhibition (§4.1.4), we spent a lot of time with the restorers and conservators just browsing through the geometry, observing the color traces and markings on the painting surface, interpreting the different shapes and details. As we will see, these observations have also been used for dissemination when creating the interactive kiosk (§4.1.4), using the same geometries to explain to visitors the different techniques used by Pollock during his work.

This operation is done mostly by observation, as the shapes must be *interpreted* by experts: the ideal situation was having curators and restorers (for their knowledge of the artwork and of the painting technique) and an expert of 3D (to help manipulate the 3D data, control rendering and taking measurements) working together. Looking at the geometry with the conservators, even in a small area like the one shown in Figure 4.4 (7x15 cm), it was easy to identify the

different traces left by the various techniques used by Jackson Pollock at work, and accurately perceive how these traces are superimposed, going over and under, creating a landscape-like impression. In this small area, it is possible to observe traces of paint squeezed out directly form the color tube (the wide flat trace with raised borders on the right), oozed out of a syringe (the smooth tear-shaped trace with the long tail), dripped from a brush or stick (the thin almost-vertical line one third on the left side), as well as the effects that Pollock used to build-up the texture and grain of the surface, like the inclusion of pebbles (the small lump near the bottom-right angle) and the corrugation of the paint due to the uneven layering (all over the image).

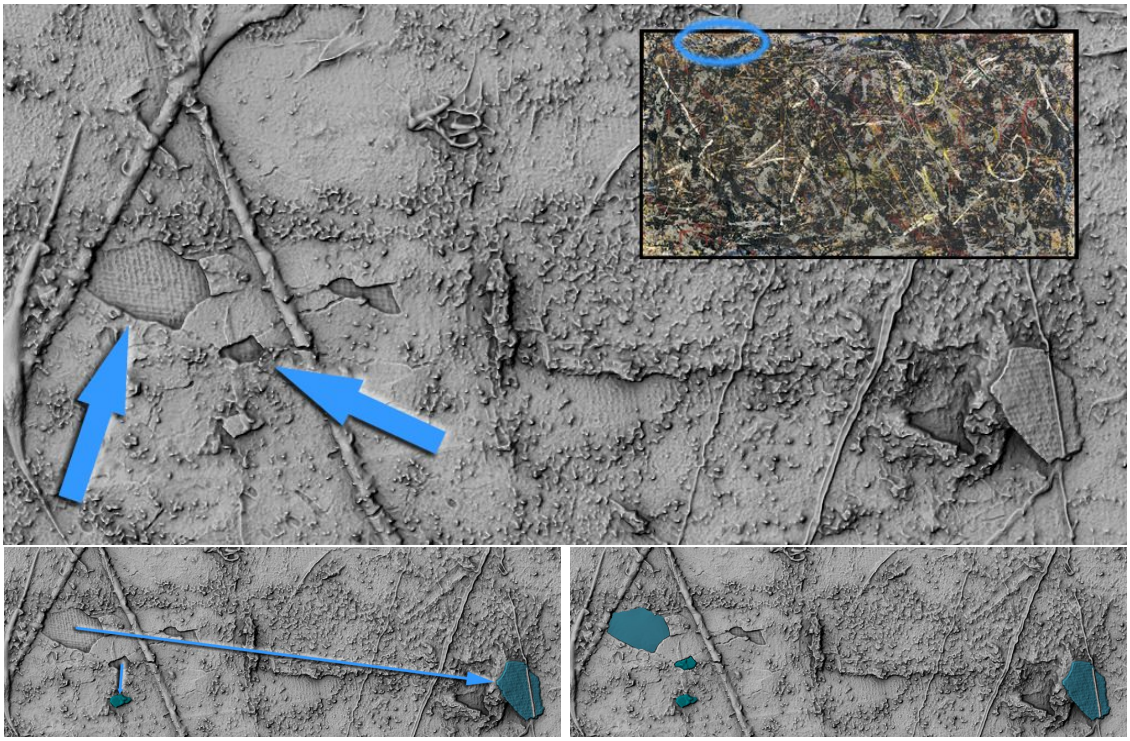

Figure 4.5: Fragments of the lead white base layer detached and re-adhered to the artwork surface during the painting process.

In some cases, the metric nature of the geometry does help in confirming the interpretation. For example, in the top-left area of the painting it is possible to see the canvas in many locations. In these small areas (few mm^2 in size), the base layer of lead white paint is detached, showing the underlying canvas. However, at a close inspection, the shape of two of the missing areas perfectly matches fragments of paint nearby. This is because the fragment detached and re-adhered nearby. Looking at the geometry, it is easy to see this strange occurrence (Figure 4.5). The interesting part is that this happened *during* the painting process, as on the re-adhered fragments it is possible to see new traces of paint.

One of these areas (the larger one) was well known by the conservators, but we found *a new one*, thanks to the increased clarity of the geometrical model. The metric nature of the model made possible an exact check, by cutting the re-attached parts of the 3D model and placing them back in their original position.

4.1.4 Publishing Action Deployment

In order to show to the public the result of the analysis in 4.1.3, we decided, together with the curators of the Peggy Guggenheim museum, to put into effect an interactive publishing action centered on the high-resolution 3D model.

This action was included in the wider framework of a temporary exhibition staged to present to the visitors of the museum the result of the cleaning and diagnostic campaign. The exhibition "ALCHEMY BY JACKSON POLLOCK. Discovering the Artist at Work" was inaugurated in the Peggy Guggenheim Collection in Venice on February the 14th 2015, and lasted until September.

To create our installation we decided to exploits the capabilities of the 3DHOP framework. Being designed expressly to handle geometries with high level of complexity, and providing a large set of customizable features to effectively map the informative layer on the 3D model and to efficiently design a user interface tailored on it, 3DHOP resulted the perfect solution for our purpose.

Moreover, since 3DHOP is a client-side system, using it we had the possibility to develop at the same time a solution that could be accessed both on-site at the museum (as a museum kiosk running on a local Web server) and on the Web (as a classic Web3D application running on a remote Web server).

Designing and Developing the Interactive 3D Viewer

The idea was to design an interactive 3D application for giving to the visitors the opportunity to freely explore the geometry as we did with the conservators and restorers.

As said before, thanks to 3DHOP the developed viewer can be experienced, through a minimal and easy to use interface, both online, acting as classic Web application, and locally, on a large touch-screen (50 inches) acting as museum kiosk (Figure 4.6). With minimal changes the 3D viewer source code was adapted to both the cases. To work on local kiosk, we used a local Web server, with the Web browser in full-screen and the OS working in kiosk mode. Minimal tweaking was made to the parameters controlling the data streaming and memory occupancy, as well as the model management:

- on the kiosk, as this is the only application running, and we know exactly how much memory we have, the cache limits were raised for a higher quality rendering;

- on the Web version, the streaming speed was tuned to avoid choking the users connection;
- the model was compressed in the Web version, to ensure faster data transfer, and left uncompressed in the kiosk, as the disk-to-memory throughput is much higher than network streaming capabilities.

Beside these changes, the two system are identical.

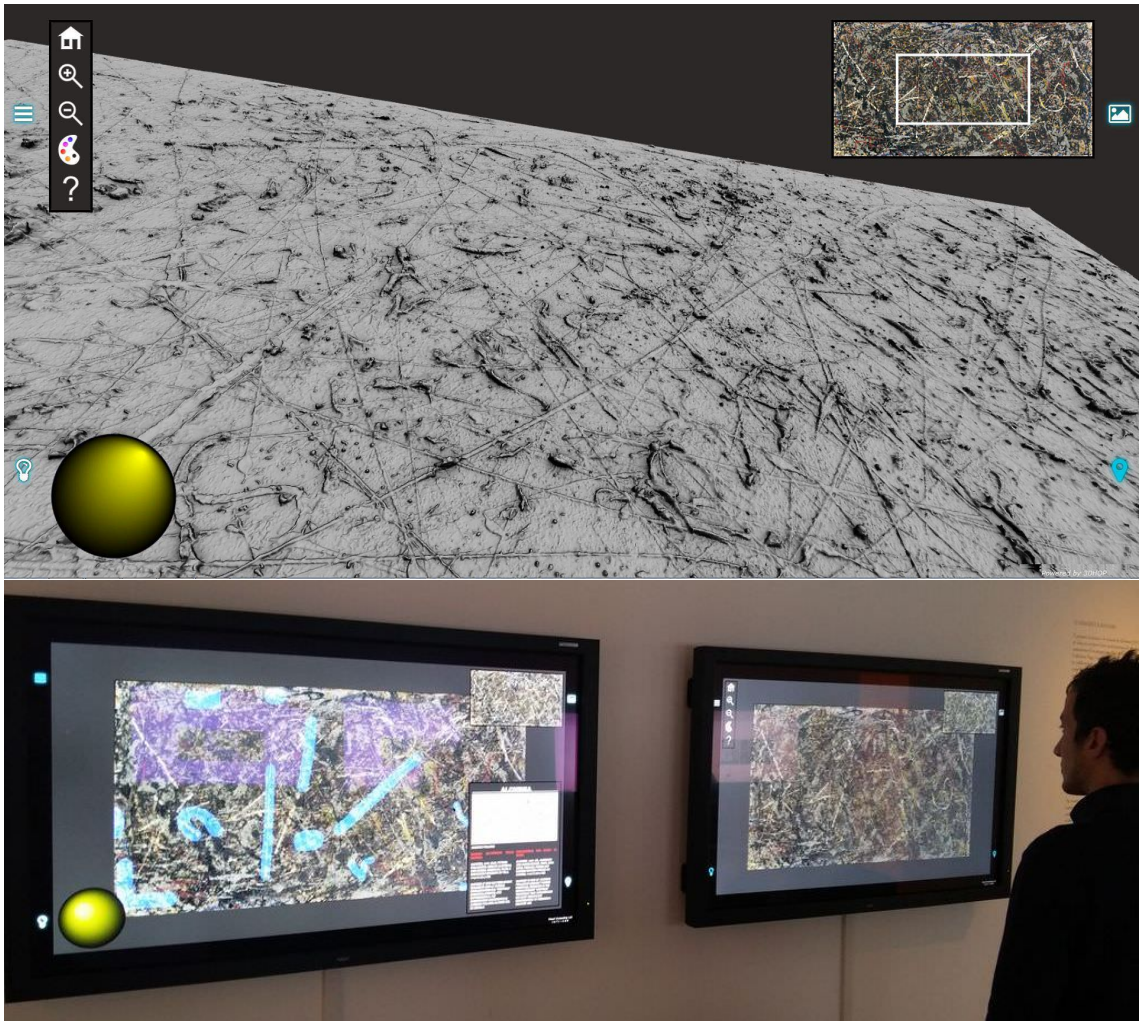

Figure 4.6: The interactive 3D viewer created for the the exhibition, running (from top to bottom) as Web3D app [Vis15c] on the Web and as museum kiosk on large-format touch-screens.

The high-resolution 3D model of the whole painting (80 million triangles) was managed using the 3DHOP multi-resolution rendering engine, to ensure interactivity and high-quality rendering in both the configurations.

The final 3D application was tailored on the needs of the specific project exploiting the 3DHOP modular structure and the JavaScript flexibility. In some case the basic features provided by 3DHOP were heavily customized. The main intervention areas were four: navigation, user interface, rendering appearance and media integration.

To improve the 3D scene navigation possibilities takes a relevant part of our work. Because of the kiosk installation requirements, we explored a lot the touch-based possibilities. The touch-screen gives the possibility to navigate the model using the interaction paradigms used for tablets and smartphones. This ensures the visitors will have a very fast learning curve. Using different modalities of touch and pinch, the user can rotate, zoom, and pan, so that every possible view position can be reached. All the touch interactions were properly mapped on correspondent mouse events, so as to obtain the same behaviors also in the Web application. The original trackball was modifies to allow the final user to "fly" over the surface of the painting, while other basic interactions were customized to better fit with the handled 3D model (the *pan* movement, for instance, was modified to try to keep the view orthogonal to the painting). The navigation, constrained to the volume of the painting, was enriched with an overview-plus-detail component (see 2.4.3): a 2D MiniMap on the upper right corner of the viewer gives reference about the camera position, and can be used as well to helps the user in controlling the navigation without getting lost.

More in general, the whole user interface of our application was modified with the aim to make the installation accessible to a wider number of visitors. Designed to be minimal just for this reason, it is composed only by four main buttons (placed at the corners of the 3D viewer window), each one connected to a retractable 2D element. The illumination tool is one of these four components. It is connected to a simple on-screen control (a 2D sphere on the bottom left corner), appearing and disappearing depending on the interactions with a light bulb icon. Through this element is possible to change the 3D scene illumination by manipulating the light direction: while very simple, this feature is able to bring out even more detail from the geometry. The interaction of the moving light with the geometry radically changes the perception of the surface, making it look more "alive" and three-dimensional.

The scene appearance was another element we particularly paid attention to, modifying the existent and adding ad-hoc new shader programs. The 3D viewer starts by showing the naked geometry of the painting, to mimic the same rendering we used with conservators, employing a non-realistic shader that enhance the perception of the geometric detail. As said, we firmly believe this visualization helps a lot in presenting and reading the painting traces, also for non-experts. Using the basic color tool provided by 3DHOP is nevertheless possible to turn on the color information (the related button is included in the home menu in the upper left corner of the viewer). However, also this feature was customized, associating to the color switch an ad-hoc shader able to bring back

photo-realism in this rendering mode.

Finally, the last area on which we focused designing and developing the viewer, was related to the integration of the informative layers. We decided to connect the most interesting information with the zones of interest using the 3DHOP geometric hot-spots (see 3.3.4). The final users can view and select hot-spots placed where the details of the captured geometry gives insights on the painting process and on the gesture of the artist. These areas, some already presented in §4.1.3, have been selected to show to the visitors some of the different techniques used by Pollock, again exploiting the clarity of the 3D digitized geometry. The hot-spots are displayed on-screen with transparent blue areas (Figure 4.7). Interacting with them the user can obtain the additional information (text and images) on the box in the lower left corner of the screen. Also in this case the default 3DHOP mechanisms were modified to get more tailored features. Indeed, to each hot-spot interaction was associated a camera animation to zoom on the selected area, and also an additional feature able to automatically hidden the hot-spot blue geometry once reached the close-in position (so allowing to the final user to appreciate the painting geometry without distracting elements).

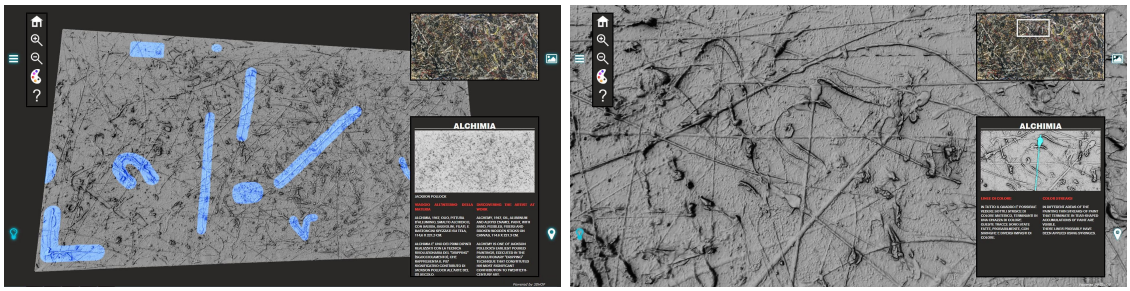

Figure 4.7: The hot-spots in the 3D viewer, each one showing relevant geometric features linked to information about a specific painting technique used by Pollock. From left to right, overview and detail (with the hot-spot geometry automatically hidden) in the Web3D app [Vis15c].

4.1.5 Results

The use of a high quality triangulation 3D scanner and a careful processing resulted in a high-resolution 3D model, integrated in the conservation process, together with a number of other diagnostic investigations.

The most interesting use of the 3D models, however, has been the work carried out in conjunction with the conservators, where the naked geometry of the artwork has been used to investigate the relationship between the shape of the painting and the gestures, actions and techniques of painting; and then to convey the resulting connections to a museum audience.

The nature of the painting and the high-resolution model made possible the discovery and the validation of several interesting aspects of the painting, in the direction of trying to reach a better understanding of the painting process which was, in the idea of the artist, an essential part of the artwork. The possibility to interactively play with the color-stripped version of the painting lets the user understand in a new way the complexity of the artwork, somehow providing an unconventional experience.

The 3DHOP framework played a central role in the disseminating these knowledges, to design and develop the publishing action. Thanks to its modular, configurable and extensible nature, it was possible to modify various aspects of the viewer to obtain a custom-tailored presentation scheme. Moreover, thanks to the multi-resolution engine, handling an extremely high resolution dataset was not an issue, and we could show to the users an astonishing level of detail.

The resulting 3D application was part of the temporary exhibition "ALCHEMY BY JACKSON POLLOCK. Discovering the Artist at Work", inaugurated in the Peggy Guggenheim Collection in Venice in 2015 and lasted eight months, totaling more than 180.000 visitors, and has also been also in a similar initiative hosted by the Guggenheim Collection Museum in New York. The Web version of the kiosk was put online at the beginning of the exhibition and it is still accessible today [Vis15c].

The results of this experience has been presented (winning the *Best Paper Award*) at the "IEEE Digital Heritage 2015 International Conference" and published here:

- *Alchemy in 3D - A Digitization for a Journey Through Matter*, Callieri M., Pingi P., Potenziani M., Dellepiane M., Pavoni G., Lureau A. and Scopigno R., International Conference IEEE Digital Heritage 2015 (DH 2015), Volume 1, page 223-231, October 2015; [1]

This publishing action also won the first prize in the category "Best Use of DH For Public Engagement" at the *Digital Humanities Awards 2015*.

The final Web3D application, a dissemination technical video, images of the work, and of the obtained 3D model, can be accessed at the project webpage [2].

4.2 Color and Gilding on Ancient Marbles: a Web3D Study

Publishing 3D content on the Web can also be addressed to more specialized aims, such as, for instance, analytic studies or documentation purposes. Following this

[1] © 2018 IEEE. Text and figures reproduced by kind permission of IEEE.

[2] `http://vcg.isti.cnr.it/alchemy`

premise, this case study [Vis15d, SPPS15] introduces a publishing action aimed to present the result of a systematic methodological process defined to detect, document, and visualize, the original color and gilding on a Roman marble sarcophagi (the Annona Sarcophagus, II-IV century AD).

The process defines a working pipeline that, starting from the selection of the artifact to study, proposes a set of investigation steps to improve our knowledge of its original painting. These steps include the direct virtual inspection, the archaeological and historical research, the on-site scientific investigation by multispectral imaging, spectroscopic and elemental analysis (eventually supported by micro-invasive techniques performed in laboratory), and the accurate polychrome surface acquisition by color calibrated 2D images.

Using 3DHOP all the data produced have been integrated with a high-resolution 3D model, to support enhanced analysis and comparisons, and even to promote a digital 3D polychrome reconstruction by virtual painting. Thus, all those data have been made accessible on the Web, using our framework to design and develop a visual media platform able to provide domain's practitioners with a new interactive 3D assessment tool.

4.2.1 Overview

Greek and Roman marble artworks have been deeply studied from a typological and stylistic point of view, while there is still a limited knowledge on the pigments, dyes, binders and technical expedients used by Roman artists. In a renewed scientific interest towards the ancient polychromy (color and gilding), a digital methodological and multidisciplinary approach can provide valuable information to better investigate and understand this fundamental aspect and to get a complete sense on Greek and Roman marble artworks.

For this reason, we present a multidisciplinary approach to identify, document and visualize the ancient polychromy on marble artifacts, starting from the analytic study of a well-defined archaeological class of artifacts with known historical period and production place: the Roman marble sarcophagi made in Rome from the first half of the 2nd century to the end of the 4th century AD.

A fundamental aspect of the proposed method is the new and innovative role of 2D and 3D digitization technologies: the production of virtual reconstructions acting as a link between the archaeological information and the scientific analyses, results to provide a better-integrated documentation (and interpretation) of the acquired data and to improve the knowledge of the original polychromy. The main idea is to integrate the computer-based technologies with consolidated scientific analyses, primarily to define a common Web3D platform to use both for a better insight of ancient color and gilding, and secondly for the dissemination of the results.

The main contributions of this 3D Web platform are:

- to propose a multidisciplinary carrier to study the marble polychrome artworks

by integration of digital technologies, scientific analysis and archaeological data;

- to promote an interactive visualization and navigation of all the acquired (and processed) data to improve their interpretation and to allow an easy and effective dissemination of the obtained results.

The Multidisciplinary Approach

In this kind of projects the publishing action is just the last step of a longest working pipeline. Usually, this final stage strongly depends by the previous steps (since the data to handle simply derive from them).

In this case study the pipeline we followed for defining the working space and selecting the dataset to gather, is based on an improved version of the multidisciplinary approach already presented in [SDC+15] and tested on the *Ulpia Domnina Sarcophagus* exposed in the Michelangelo's cloister of the Museo Nazionale Romano Terme di Diocleziano in Rome.

Figure 4.8: The so-called Annona Sarcophagus, Museo Nazionale Romano Palazzo Massimo.

The working pipeline starts from the selection of the artifact to study: in our case the so-called Annona Sarcophagus (Figure 4.8) in Museo Nazionale Romano Palazzo Massimo (inv. no. 40799), dated at the last third of 3rd century (270-280 AD). This rectangular sarcophagus without lid is decorated only in the frontal part with allegorical characters: *Portus*, *Annona*, *Concordia* (behind a married couple making a *dextrarum iunctio*), *Genius Senati*, *Abundantia*, and *Africa* (from the right to the left of the sarcophagus). It was selected because presents several visible color and gilding traces and poses interesting open problems on their application techniques.

Moreover, the academic literature reported that the sarcophagus showed a great amount of red and gilding.

After a mandatory preliminary stage in which the selected subject is thoroughly studied (via an archaeological and historical research and a direct visual inspection), the pipeline contemplates the effective data acquisition. This step includes the identification of the areas where to perform non-invasive analysis (like multispectral imaging, spectroscopic and elemental techniques), and also where micro-samples can be eventually taken to do more accurate laboratory analyses.

The acquisition stage is completed by an accurate polychrome surface mapping through color-calibrated 2D images. High-resolution and calibrated images of the sarcophagus were acquired using a Nikon DSLR camera and a color calibration chart (a Macbeth X-Rite ColorChecker Passport), taking more attention in the captures of the representative polychrome and golden areas.

Finally, the full analytical dataset (Fourier-Transform InfraRed spectroscopy-FTIR, X-Ray Fluorescence-XRF, etc.) is processed. This includes the examination of the eventual micro-samples by Optical Petrographic Microscopy (OPM) and Raman spectroscopy, in order to extract further information. Where the situation is more complicate, can be also performed an additional analysis with Scanning Electron Microscope and Energy Dispersive X-Ray Spectrometer (SEM-EDS).

4.2.2 3D Data Acquisition and Processing

The 3D model of the studied sarcophagus plays a central role in this case study, since it represent the carrier on which to map the gathered data (in addition to being the main pillar of the Web publication).

For the 3D digitization of the artifact, we decided to test and use the multiview-stereo 3D reconstruction approach. For this reason, we captured a specified set of images that guarantee an optimal sampling of the surface from different point of views.

3D Processing

For the 3D digitization of the sarcophagus we employed a multi-view stereo approach using a set of 165 photos. The images were processed using Agisoft PhotoScan [Agi10] to compute the camera parameters and a dense point cloud.

Since the processing of the whole set of images produced an incomplete point cloud, we split the images in several set corresponding to the main parts of the sarcophagus (the front, the interior and the sides). For each set we perform a separate processing, producing different point clouds.

Each clouds was triangulated using the Screened Poisson Surface Reconstruction algorithm [KH13] in order to make easier the alignment of the different parts inside MeshLab [Vis05]. The aligned triangular meshes have been merged together with the

standard volumetric algorithm based on the Marching Cube available in MeshLab, producing a final high-resolution model of 72 million triangles.

Finally, the color images have been projected and integrated on the high-resolution model to document the current state of the color and gilding using the algorithm in [CCCS08] and available in MeshLab.

Even though the 3D model presents some imperfections in the area where it was difficult to obtain an optimal multi-view acquisition (i.e. that regions hidden by high-relief part), the final model shows how the image-based 3D reconstruction can be a good and cost-effective alternative to active 3D scanning techniques.

Digital Polychrome Reconstruction

An interesting contribution of this case study (also central element in the Web publication) is represented by an attempt of digital polychrome reconstruction.

For the selected sarcophagus we propose a preliminary virtual polychrome version of the *Africa* personification (Figure 4.9) with the purpose to show the distribution and overlapping of the colors on the sarcophagus surface, without any simulation of the light interaction with the painting that requires more complex reflectance functions (BRDF, SVBRDF, BSSRDF) and rendering systems [DC08]. It is partially reliable due to the lack of clear clues on the original color of some elements that need additional scientific and archaeological investigation.

Figure 4.9: Hypothesis of 3D polychrome reconstruction in the Annona Sarcophagus, visualized through the developed Annona Web3D application [Vis15d].

The polychrome reconstruction has been made using the painting tool of MeshLab, exploiting the RGB coordinates identified in the most representative color traces in the set of calibrated images acquired in the previous step. In the

identification process of the RGB coordinates of each color, we took multiple samples in different measurements points and we noted very similar RGB coordinates across the multiple samples of the same color.

A fundamental step for the polychrome reconstruction is the integration of the data acquired and computed in the previous stages (see §4.2.1). In the specific, the Visible-induced IR Luminescence (VIL) and UltraViolet-induced (visible) Luminescence imaging (UVL) analysis (multispectral imaging techniques based on photo-induced fluorescence) are useful in the characterization of the spatial distribution of Egyptian blue pigment and organic dyes, such as the rose madder lake. The OPM and Raman spectroscopy identify the inorganic pigments and the organic dyes. They are also useful to determine if a color is made of a single pigment (with binder) or a mixture of more pigments and dyes. Therefore, the OPM can reveal a possible stratigraphic sequence and if the color is applied directly on the marble surfaces or over a fine ground layer.

Finally, the scientific results, connected with the direct visual inspection and archaeological data, are very useful to reproduce a proper shading of single or overlapping colors and to propose a painting style as similar as possible to the original.

4.2.3 Publishing Action Deployment

Having the possibility to combine different diagnostics modalities with archaeological data is very interesting for the specialists of polychromy. However, usually the integrated comparison of scientific and archaeological data is one of the most complex process done by the researchers (or conservators) while studying an archaeological artifacts. A qualitative integration of multidisciplinary data is often an issue, as well are accessibility and visibility related to the outcomes. The publishing action, final stage of the methodological approach followed in this applicative experience, just aims to solve these issues.

Designing and Developing the Interactive 3D Viewer

Our idea was to design a 3D Web platform centered on the generated high-resolution 3D model, which would let us to exploit the digitized artifact as main informative channel for the mapping of the gathered analytical data and of the digital 3D polychrome reconstruction hypothesis.

The main goal of this Web3D platform would be to give to a specialized audience the opportunity to freely interact with the artifact geometry, studying and interpreting the multidisciplinary information proposed, without preventing to less skilled users to improve their knowledge on pigments and dyes used by Roman artists, or even on the pictorial styles and techniques used to apply both color and gilding.

Figure 4.10: The Annona Web3D viewer [Vis15d] interface with the rendering of the 3D multi-resolution model of the Annona Sarcophagus.

We used the 3DHOP framework for designing and developing this application (Figure 4.10). As usual, the advantages for using the 3DHOP framework in this kind of applications are several: starting from the support to the efficient visualization of very large 3D models also on commodity computers and standard internet connections, through the easy integration of all the types of data gathered in the 3D scene, up to the user friendly interactive exploration of the 3D scene.

Differently from the example in 4.1, in this case study we didn't develop any ad-hoc component for 3DHOP; conversely, to build the aimed platform we mostly experienced the default elements provided by the base framework. We mainly focused on the effective integration of the wide scientific analytical dataset, at the end customizing just the 3D viewer user interface: a relevant component in making the heterogeneous set of informative layers easily accessible to the biggest number of final users.

The 3DHOP features of the final documentation system includes basic framework's elements, exploited for implementing the interactive navigation of the 3D multi-resolution model (with the possibility to freely examine any details of the artifact using the zoom and pan commands), or for enabling the possibility to interact with a directional light (that can help to highlight some details of the relief otherwise not discernible, like for example small inscriptions).

At the same time, three more elaborate elements were added to the viewer interface for better integrating the multidisciplinary informations. Each one of these components, useful both for the analytical process and the dissemination,has been designed with a functional interface composed by a button plus a customized drop-down menu.

The selector for the interactive visualization of the scientific polychrome reconstruction hypothesis is one of those. More specifically, by clicking on the palette colors icon (Figure 4.9) the user can switch from the current color to the original virtual color proposal (or more presumed reconstructions). This element exploits the instance visibility control provided by 3DHOP (see §3.3.4) to manage the scene composition switching between two different geometries (one for the current 3D model and one for the reconstructed version).

The second additional component concerns the management of the multiple levels of information. A set of geometric hot-spots (§3.3.4) are exploited to link the different analytical data to the 3D model. The hot-spots are subdivided in different groups that can be visually identified by using different colors. Each group correspond to a different acquired data (e.g. VIL and UVL imaging, OPM, Raman spectroscopy, etc.). The full list of the scientific analysis is visualized in a dedicated drop-down menu that allows the user to choose among them (Figure 4.11). To each hot-spot there is the possibility to associate a pop-up window that allows the user to consult the hot-spot data content.

Figure 4.11: Annona Web3D viewer [Vis15d] showing the Annona Sarcophagus areas where the VIL analysis were performed.

Finally, exploiting the 3DHOP trackball automation features (again introduced in §3.3.4), a "storytelling" component was also added to the presentation. It allows a guided navigation using a predefined set of views related to most interesting details (more preserved color area, polychrome details useful for the virtual reconstruction, elements that attested particular events like an ancient or modern re-painting/restoration). Also in this case the final user can access to the list of the bookmarked views through a dedicated drop-down menu (Figure 4.12).

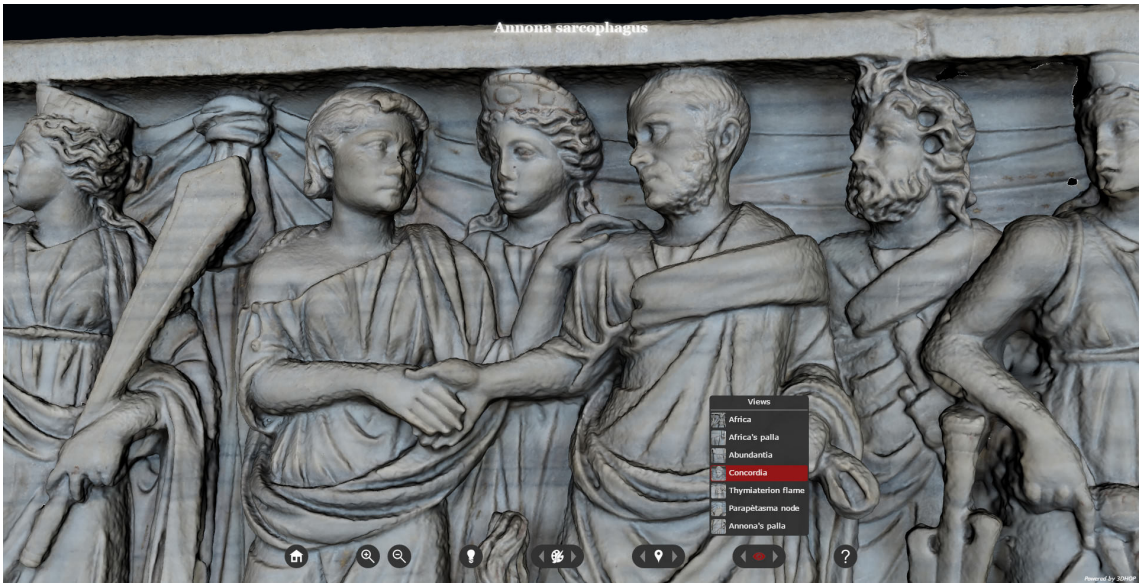

Figure 4.12: The Annona Web3D viewer [Vis15d] enables the definition and visualization of a predefined set of views, selected to show the most representative polychrome areas on the Annona Sarcophagus.

4.2.4 Results

The integration (and visualization) of the archaeological data and the scientific analyses results over a high-resolution 3D model represents a novelty in the study of the ancient polychromy and allows significant improvement for the dissemination of the knowledge gathered. Indeed, new results and better understandings of Roman sarcophagi are (and can be) obtained by linking together, and comparing, digital datasets gathered with several techniques.

In the presented case study, the digital 3D model of the Annona Sarcophagus (MNR-PM, inv. no. 40799) was used as main carrier where to map the scientific examination done and the novel elements/techniques detected. We have showed how the 3DHOP framework could be effectively used (exploiting default features or modifying the existing components) to enable a technical presentation, able to support the visualization and the dissemination of all the collected (raw and processed) datasets on the Web, once those data are mapped on a high-resolution 3D model.

At the end of this analytical and multidisciplinary process, the published 3D geometry has also used to propose a preliminary hypothesis of a scientific polychrome reconstruction. This virtual reconstruction shows the distribution on the surface of the main identified colors, highlighting the potentiality of the proposed digital system and multidisciplinary approach.

Our aim was to exploit 3DHOP to propose a novel tool allowing domain specialists to give visibility and ubiquitous access to these information, helping

them in publishing their work. From this point of view, the final solution [Vis15d] seems to have reached the goal. 3DHOP was able to manage the high-resolution data of the project, allowing the user to access the maximum possible detail. At the same time, it was possible to configure the user interface to integrate, in a structured way, all the diagnostic and scientific highlights, to build what is, essentially, an interactive version of a scientific report.

The results of this experience has been presented at the "IEEE Digital Heritage 2015 International Conference" and published here:

- *Digital Study and Web-based Documentation of the Colour and Gilding on Ancient Marble Artworks*, Siotto E., Palma G., Potenziani M. and Scopigno R., International Conference IEEE Digital Heritage 2015 (DH 2015), Volume 1, page 239-246, October 2015; [3]

The final Web3D application, a demonstration video, and additional images of the work, can be accessed at the project webpage [4].

[3] © 2018 IEEE. Text and figures reproduced by kind permission of IEEE.
[4] `http://vcg.isti.cnr.it//roman-sarcophagi/annona-sarcophagus`

Chapter 5

Evaluating 3DHOP in Developing Web3D Publishing Services

In Chapter 4 we have discussed two publishing examples, based on real experiences, aimed at validating, in a comprehensive and accurate way, the design choices taken during the deployment of 3DHOP. However, those case-studies, despite aiming towards different communication needs (public dissemination and support to scientific analysis), were both focused on the direct publishing of single, specific datasets. Conversely, in this Chapter, we would like to prove the effectiveness and flexibility of 3DHOP when working in the contexts of assisted, automated, and bulk publishing.

For this reason, we will describe two others relevant testbeds that, again, differing in their nature, are still both characterized by the basic idea of non-direct publishing: providing a service for the easy publication of complex multimedia assets, and proposing a pipeline useful to rationalize and speed-up the publication of heterogeneous 3D dataset on a Digital Library (DL).

More specifically, the first contribute can be contextualized in the ecosystem of the assisted publishing services, and describes the steps we did in designing and developing a platform that would make extremely simple to publish and share online complex data (3D content and other specialized data). The target audience is, again the Cultural Heritage community. In this case-study, the core of 3DHOP has been used as code base for supporting the server based visualization structures, but also as template for designing the components aimed at handling other two visual media: 2D Hi-Res Images and RTIs. The resulting platform (called ARIADNE Visual Media Service) is an automatic service able to transform the uploading of these media file into an efficient remote visualization on the Web.

The second contribution, instead, examines the strategy developed for publishing online an extensive and heterogeneous set of 3D digitalizations, with the final goal to build an enriched multimedia Web repository. In this example 3DHOP has been used as pivot for the design of a complete pipeline, playing a key role both in the development of the data pre-processing stage, and in the building

of the final Web3D viewer. This experience resulted in a real DL (the Edition Topoi Collection) enclosing thousand of 3D models, freely accessible in an interactive way and effectively integrated with other media informations.

Both the presented experiences, ranging from assisted to automatized publications, have been characterized by a common approach focused on the service-based and data-oriented path of the publishing actions. In these two cases, the developed solutions have exploited the "innermost layers" of 3DHOP (page creation, data structures, scene creation and management), more than its interface and interaction capabilities, making it possible to explore and validate the flexibility and the adaptability of these parts of our tool.

Once again, the outcome provided by these two real-world experiences seems to confirm the design choices detailed in 3.

5.1 ARIADNE Service: Easy Web Publishing of Advanced Media

The possibilities to manage on the Web some kind of very specialized media assets are very limited. The intrinsic complexity of these contents, paired with the lack of technical, but still accessible, publishing solutions, makes their presence on the Web quite rare. In several scientific domains, where nowadays it is mandatory using together several of those technical datasets, and the vast majority of practitioners is not trained and not used to handling complex web-publishing tools, this issue seems to be more critical.

In this specific project [Vis15b, PPD+15], we tried to develop a platform aimed at facilitating the building of Web presentations, specialized on a CH oriented selection of these specialized media. Our idea was to design an approach able to transform, through few and simple steps, an uploaded media file in a remote interactive visualization on the Web.

The ARIADNE Visual Media Service (Figure 5.1) is the tangible result of this idea. It is a service-oriented platform, based on a simple Web interface, addressed to assist the deployment of specialized Web presentation of three different visual media. For the development of this platform we used the 3DHOP framework has a code base, increasing its degree of specialization in efficiently handling specific data assets, but, at the same time, simplifying the publishing paradigm to make it accessible to a wider range of content creators.

This activity has been designed as part of the European Project project ARIADNE [Vis15b], a network project focused at bringing together and integrating existing archaeological research data infrastructures. In this framework, the Visual Media Service can be seen as one of the various interconnected services provided by the project to the archaeological community.

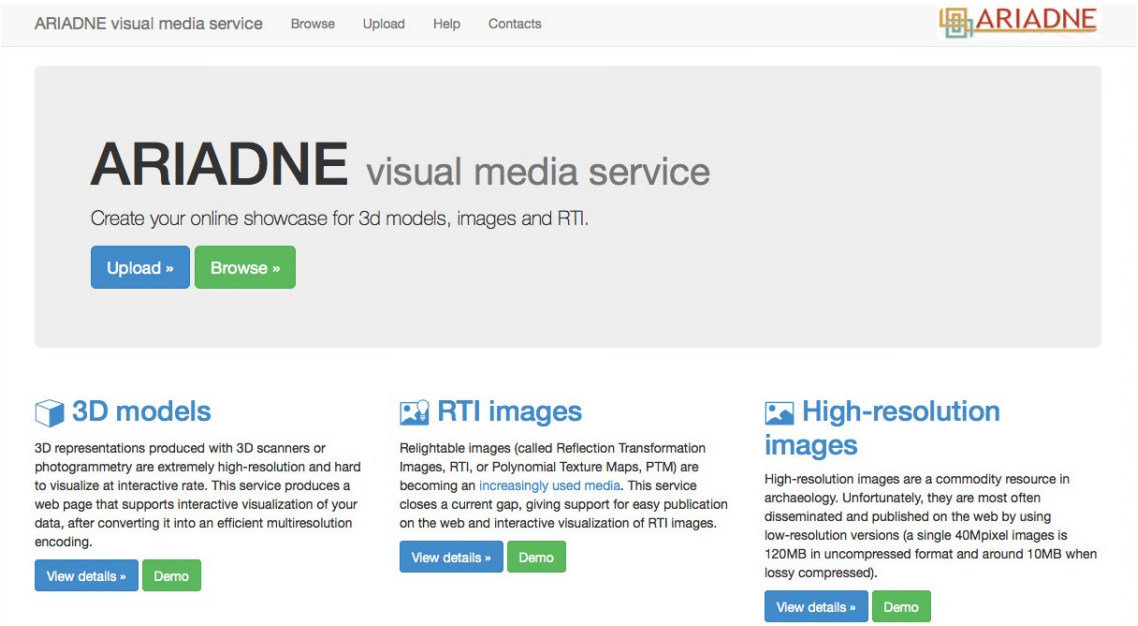

Figure 5.1: The ARIADNE Visual Media Service [Vis15b] homepage.

5.1.1 Overview

In the CH community, with the term visual media we encompass any data that could help to better represent, document, and communicate, an artworks under investigation or study. Visual media are not new instruments of work in CH, since drawings and images have been used for centuries, and they are part of the common working practice. The new issue, for domains like CH, is how to make a proficient use of those media, especially now that different digital incarnations are made available by the progress of the technology [SD13].

Web publishing could be a sensible answer, but for this field publishing visual media online is not always that easy. Practitioners usually not able to use complex/specialized tools, and poor availability of solutions aimed to unskilled content creators, often are main issues. An additional critical aspect is connected to the same visual media to handle, generally constituting a very heterogeneous set (ranging, for instance, from 2D HDR images to high-resolution 3D model) of richly annotated (through barely standardized methods) data intrinsically complex (that often not allows quality degradation, because of peculiar domain needs).

As we have seen in Chapter 2 for the 3D case, nowadays a lot of Web publishing systems are available. However, most of these solutions are addressed to mainstream media publishing, and are unable to handle specialized technical dataset. In the best case, the few specialized solutions reasonably accessible are focused just to a single media, so requiring to multiply the learning curve for each one of the aimed publications, and offer uneven access to multiple media types.

Conversely, our approach has been to design a service that allows untrained users

to upload multiple types of visual media file and, after some remote processing, to simply receive the URL where the specific data has been stored and could be accessed for the interactive visualization. Moreover, the system also offer the possibility to download the processed data and interactive webpage, making it possible for the user to integrate the visualization page inside its own website, avoiding to rely on external repositories.

The system has been designed to be lightweight and simple to use: we decided a minimal set of metadata useful to characterize the final webpage; we developed a basic Web interface aimed at assisting the publishing process; and we strove to provide a uniform browsing experience for the various media types.

5.1.2 Which Type of Visual Media?

The concept of visual media can be referred to all those assets characterized by a visual representation. Referring to the professional CH world, in this group we may find: 2D "basic" images (standard photos, very-high-resolution images, photomosaics, orthophotos, geo-referenced images, and so on), 2D+ "specialistic" images (high dynamic range, panoramic, multispectral, re-lightable images, and so on), 3D content, and videos (again, multiple kinds: panoramic, HDR).

Nowadays, a number of new, low-cost, and commodity opportunities for the easy acquisition/generation of these visual supports are available and largely used on the field. Current sampling technologies allow the production of very complex and high-resolution representations (up to Giga-pixels or Giga-vertices).

Applied to the specific CH domain, those digital models are exploited to cover a very wide scale extent, going from representations of small findings (few centimeters) up to representations of an entire archaeological site (hundreds of meters). These supports allow representing faithfully even small regions or subtle details of the object of interest; therefore, the massive use of sub-sampling and compression (in the case of images), or geometry simplification (in the case of 3D models), is usually not proper (especially for CH applications).

As we said before, how to open those specialized data to the external world (considering both experts and common people), and how to publish them in an easy and efficient manner, is still a critical issue. The bigger problem is related the lack of tools and experiences for sharing and publishing on the Web those visual media, either as independent resources or as part of structured archives. A very successful Web approach has been the one endorsed by YouTube, providing a simple interface for video data upload, and masking to the user all the processing needed to convert and post those data on the Web. Sketchfab [Ske14] has endorsed a similar approach, but just for 3D models and mostly focusing on the low-resolution end of the spectrum.

The goal of this experience was to provide an online publishing service focused on a selection of those media, that could provide an accessible service for CH professionals and their hi-resolution data. We chose to manage three different

media types in our server: high-resolution 2D images, RTIs (Reflection Transformation Images, i.e. dynamically re-lightable images), and high-resolution 3D models. Choosing contents mostly characterized by a strong accent on the *high-resolution* feature, our idea was to cover a "market share" that was not addressed by other services and tools.

Figure 5.2: A visualization webpage automatically generated by the Visual Media Service [Vis15b] for browsing a high-resolution 2D image.

High-Resolution Images

Images are the more common visual media, and they have been part of CH datasets right from the very beginning, originally by means of the analogical, printed version and more recently by means of digital supports (either digitally native images or scanned from old prints/slides). Those data are part of most digital archives and collections.

While images are a medium that is fully integrated with the Web and HTML since its birth, there are a few aspects that lack of a standard solution for archival and visualization purposes. Most of the images produced nowadays are very high-resolution. High-resolution images are now a commodity resource, with the impressive evolution of digital photography (just to mention a single example, a recent off-the-shelf smart-phone provides a 41 MPix camera) and the wide availability of tools that allows aligning and stitching image patchworks, allowing users to reach huge image resolutions.

When high- or huge- resolution images are available, the visualization on the Web can be difficult, due to the amount of data that have to be transmitted before the Web browser would be able to present visually something. This is because standard Web browsers have to receive the entire file before visualizing it.

A possible solution to avoid the problem of waiting the entire image transmission before being able to provide a visual feedback is the approach adopted by maps viewer. For example, Google Maps [Goo11b] handles its huge maps by encoding them in a sequence of decreasing resolutions. Each image of this sequence is split up in square tiles of fixed size (usually 256 pixels per side) to allow for the data management at high granularity. The client in the browser "composes" on the fly the portion of the image selected by the user using the tiles more suitable, according to the size of the portion under view. This approach is based on a simple multi-resolution encoding that has been demonstrated to be very efficient to visualize this type of data. A similar approach can be employed to visualize high-resolution images, based on tiling and hierarchical representation schemes.

Figure 5.3: The visualization webpage automatically generated by the Visual Media Service [Vis15b] for browsing an RTI.

Reflection Transformation Images (RTI)

Re-lightable images (more technically called Reflection Transformation Images, shortened as RTIs) are becoming an increasingly used technology to acquire a detailed and interactive documentation on quasi-planar objects

([MGW01, PCC+10]). This is particularly useful especially for objects characterized by a complex light reflection attributes. The advantage of this representation is the possibility to change the light direction over the images in real time (i.e. at visualization time), and the availability of using enhanced visualization modes to better inspect fine details of the objects surface.

RTIs have been successfully applied in a number of applications, such as collections of coins, cuneiform tablets, inscriptions, carvings, bas-reliefs, paintings and jewelery. Moreover, RTIs enable the digital representation of artworks made of materials that cannot be easily acquired by usual 3D scanning technologies (highly reflective materials, semi-transparent object, etc.).

Typically, this type of images is generated starting from a set of photographs acquired with a fixed camera (positioned on a tripod or an acquisition gantry) under varying lighting conditions. RTI encodes the acquired data in a compact way, using view-dependent per-pixel reflectance functions, which allows the generation of a relighted image using any light direction in the upper hemisphere around the object location. This per-pixel reflectance functions vary between different RTI types. For PTM (Polynomial Texture Maps) the function is a bi-quadratic polynomial (6 coefficients are required to define it). For more advanced RTIs, Hemi-Spherical Harmonics (HSH) are usually employed (9 coefficients are used in this case). In this last type of RTI, the image is subdivided in nine layers, one layer for each HSH coefficient, where the i-th layer contains the i-th coefficient of the three RGB color channels. Then for each layer a multi-resolution quad-tree is created and a tile for each node of the tree is saved in JPG format. To visualize a specific pixel, we need to load the nine JPG images that contain its HSH coefficients.

The interactive visualization of RTIs can be supported locally, using freely available tools. On the Web, it is possible to use WebGL to manage the rendering of the RTI in realtime with a shader. However, as just stated for high-resolution images, since usually RTIs may be at large resolution, also this visualization component should adopt a similar tile-based hierarchical approach.

3D Models

As we said before 3D representations have become quite common in CH. In particular, sampled models produced with active 3D scanning (laser-based systems or systems using structured light) or adopting the recent photogrammetry approaches (production of 3D models from set of 2D images), are more and more gaining momentum in this context, since those models are a scientific and detailed representation of a real CH artefact.

Nowadays, there is a pressing need for platforms supporting easy publication of these models on the Web, mostly aimed at these sampled/digitized 3D models. These complex data structures, generally rich of informative content, cannot be confined to the single local archive, but should be shared with the community, to increase knowledge and stimulate further study.

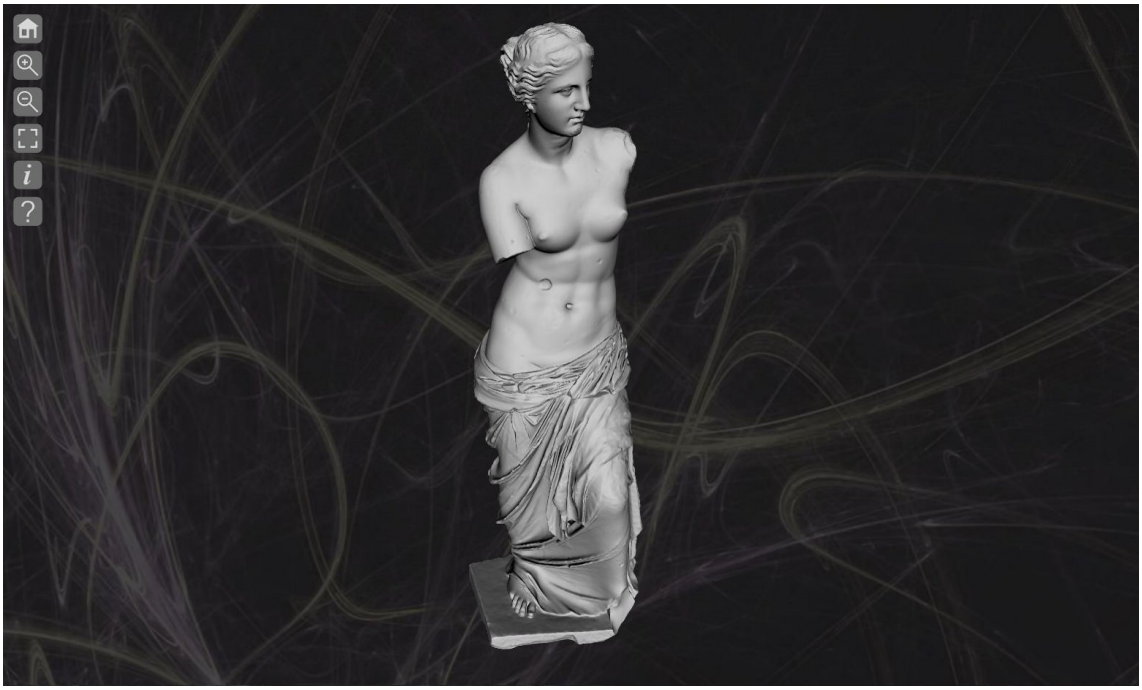

Figure 5.4: The visualization webpage automatically generated by the Visual Media Service [Vis15b] for browsing a high-resolution 3D model

As we have seen in Chapter 2 and 3, presentation on the Web of sampled 3D models is still a difficult task to achieve, and, despite the Web3D panorama is plenty of smart solutions, currently only few of them are implemented as publishing services for this kind of data.

5.1.3 Publishing Service Deployment

In order to ease the online publishing of the assets just introduced in 5.1.2, we wanted to provide a service where the content creator does not need any knowledge of the issues related to visual media publication on the Web (handling of the different file types, compression/multi-resolution streaming and rendering, etc.).

Some of the specifications of the systems, in terms of functionalities and access, come from the European Project project ARIADNE [Vis15b]. In the framework of this project, the Visual Media Service was just one of the services of an integrated network of supporting tools for collections of data and metadata. This is the reason why, in our service, are not enforced strong concepts for “users”, “galleries” and “collections”: the idea was that the project superstructure would be used to interface with the underlying services, filling these roles. The same goes for the management of the metadata associated to the uploaded media: indeed, to provide a more structured access to the service data it is delegated by the project superstructure. However, for testing purposes, and since we believe that the system could be helpful for the

community, we have decided to open up the service to all users, even if not coming through the ARIADNE project infrastructure.

Designing and Developing the Web Service

The uploading front-end of the service is based on a simple form (Figure 5.5), where the user upload its own data (3D models, hi-resolution images or RTIs) and to provide some basic information about the file and the represented artwork.

As a first step, the service processes the input data in an automatic way, so as to transform them in a Web-friendly multi-resolution format, that will grant easy and efficient access and remote visualization on the Web. Then, the system will generate a webpage containing an interactive viewer properly built around the specific media type and content.

At the end of the pre-processing and format conversion stage (that is asynchronous, as its extent, depends on the complexity of the uploaded data), the user receives an email containing a link to the visualization page (hosted on the Ariadne Web-service and open to any external user) and to an "admin" page, where the user can modify the associated metadata, or even delete the data and its viewer. From this page, It is also possible to download the created page (HTML code + processed 3D model or 2D image) in order to store and integrate the content on a users local server or archive. The user (at uploading time or from the admin page) may also decide to keep its data private: in this case, the viewer will not be listed in the "browse" section of the site, and it will be accessible only if the user provides the direct link.

These choices also aim at providing multi-level access to the system. The most basic of user will just upload a media file and leave the visualization page on the Web service server, to share it with colleagues or link it from its own pages. A more advanced user may download the viewer and integrate it inside its own webpage. A user with programming skill may decide to use the created webpage as a starting point for the development of more complex applications (taking advantage of the advanced features provided by 3DHOP).

The core of the 3DHOP framework has been exploited in the development of the platform back-end. Its basic structures, already able to efficiently handle high-resolution 3D models, has been extended to manage also high-resolution 2D images and RTIs. The philosophy followed in the addition of these new assets has been to mimic as much as possible the basic interaction implemented for the manipulation of 3D geometries, so as to have structural coherence and consistency.

High resolution images are transformed into a multi-resolution format, supporting progressive streaming: the high-resolution image is regularly divided in tiles and a hierarchy of images at different resolutions is produced from these tiles (the process is similar to the approach adopted in Web maps applications). In the visualization page, a WebGL-based rendering component will fetch the appropriate

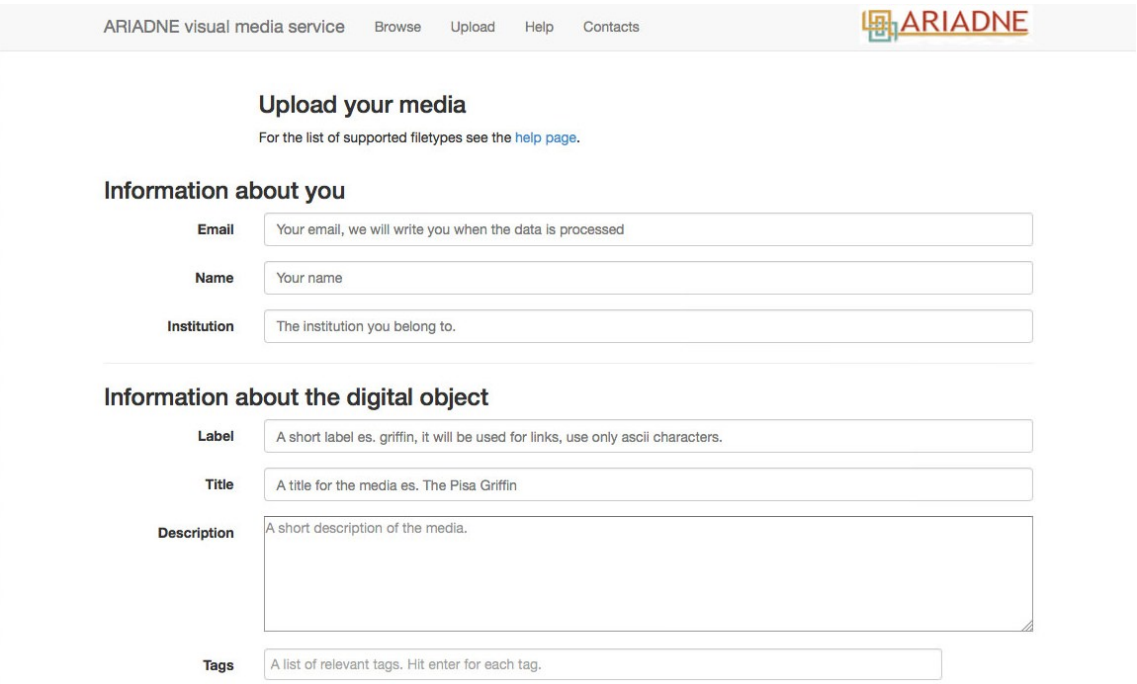

Figure 5.5: The ARIADNE Visual Media Service [Vis15b] upload page.

chunks of images from the server, and use them as textures, providing a smooth pan-zoom navigation (see an example of the image browser in Figure 5.2).

RTIs are managed similarly to hi-resolution images, even if the encoding for the Web streaming is a bit more complex (due to the multiple coefficients needed to represent the behavior of each pixel). The WebGL rendering also takes care of the user input and calculation of the variable-lighting (see an example of the RTIs viewer in Figure 5.3). The way we process and encode RTIs to provide Web-based visualization is described in details in [PSP+12].

In the case of 3D models, the geometry is pre-processed automatically converting the 3D model into the multi-resolution format Nexus and then compressing it (detail on ther multi-resolution engine has been already introduced in §3.3.1). The visualization webpage built using 3DHOP, an example of the browsing page for 3D models is presented in Figure 5.4.

As said, all three media heavily exploit the concept of multi-resolution. This idea, for Web-based access of high-resolution data is ideal for a number of reasons:

- minimize the CPU usage, as the assembling algorithm is quite simple. This is especially important since the client side is developed in JavaScript;
- using a collection of fragments naturally supports an out-of-core approach,

which allows us to start rendering as soon as some data is incoming, and let tile-based data processing to minimize the effects of the network latency;

- it is possible to optimize the rendering quality for a given bandwidth value;
- automatic pre-fetching is implemented to hide latency as much as possible;
- there is no need for special server support: data transmission just requires the basic HTTP protocol (in other words the browser itself handles both the data streaming and rendering tasks).

The last point is particularly important. Indeed, not requiring a specialized server infrastructure, the deployment of this online viewer became much more approachable by inexperienced users.

Finally, the data structures for remote visualization (multi-resolution for 3D models, image pyramids for images and RTIs Web encoding) also provide, as we already said, a very basic protection for the original data, since the direct download of the multimedia encoding in a single, plain file is not possible. As the data is streamed and accessible, it is always possible, with some effort, to access the whole dataset, but require skills beyond the basic.

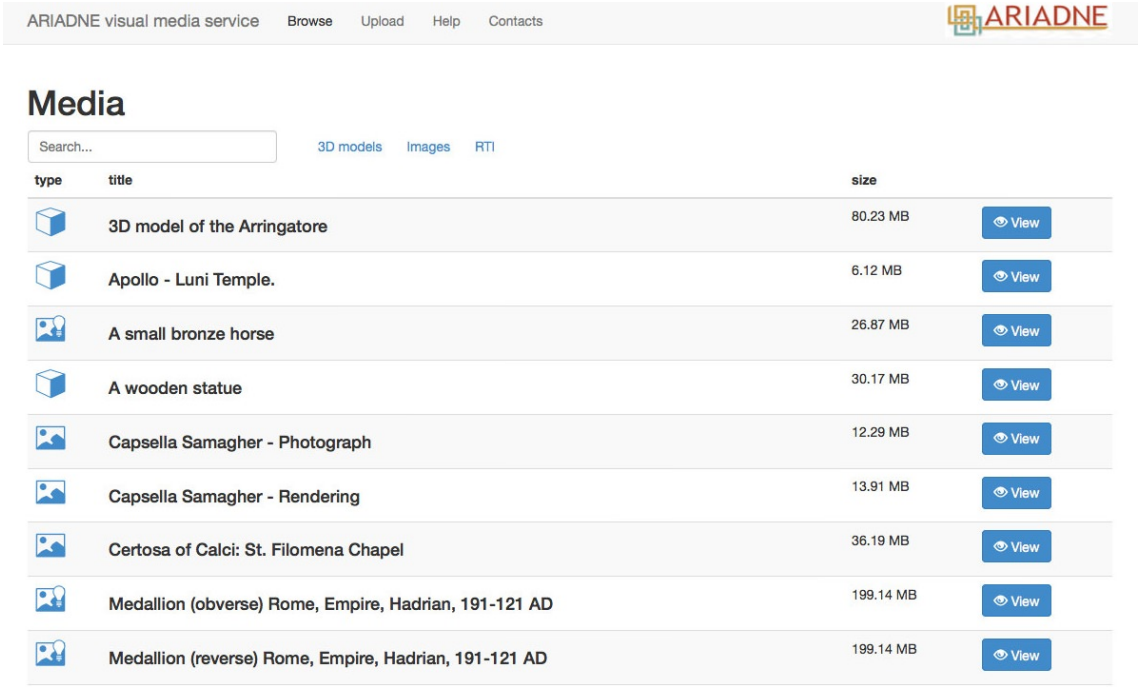

Figure 5.6: The ARIADNE Visual Media Service [Vis15b] browsing page, that provides access and searching over the uploaded visual media data.

5.1.4 Results

The ARIADNE Visual Media Service represents an answer to the need of using and sharing online complex and technical media types in the context of Cultural Heritage. Thanks to this platform, inexperienced users can easily upload different visual media files getting back a ready-made interactive Web visualization of them. Differently from the existing solutions, the system developed in this case-study implements an assisted publishing paradigm addressed to specialized set of content composed by several media assets.

This experience explores the 3DHOP possibilities as a code base for developing a service-oriented application. Integrating 3DHOP in this project was very simple. As all the creation of the scene and the configuration of the interaction and behavior is done using a declarative approach, it is easy to create a 3DHOP viewer from a script, populating the fields and parameters using data stored in the server. As 3DHOP can be configured in order to use different manipulation tools (trackball) and the different tools are modular, it was also possible to add a degree of customization to the viewer page, to cope with the specific needs of the dataset. The other viewer components have been somehow uniformed to the 3DHOP behavior and interface, in order to provide a seamless experience to the user. Introducing a better degree of customization also for high-resolution images and RTIs would result in a more tailored experience for the user, and it is certainly on the roadmap.

Thanks to 3DHOP flexibility the ARIADNE service results to be accessible at multiple levels, since it allows to more skilled users to exploit the resulting viewer for different uses, for instance using it for embedding media content in external webpage or as a base for creating more structured presentations. Concerning this last point, if we limit the discussion just to 3D datasets, the ARIADNE service can be also seen as an entry system for approaching the 3DHOP use (since content creators could get a basic 3DHOP viewer using the ARIADNE platform, and then start to modify it adding on top of it advanced 3DHOP features).

Initial testing of the platform, started on selected datasets provided by ADS [Uni96] and Discovery Programme [Her91], led to a positive user evaluation. However, since the ARIADNE Visual Media Service is linked to different Europen Community projects (the project ARIADNE [Vis15b] before, and the project PARTHENOS and DARIAH now), with a good number of users daily uploading their dataset and populating our server (in Figure 5.6 a sample of public media currently uploaded in the ARIADNE media server), the validation of the platform is still an ongoing process that continues day by day.

The number of project in which the platform is involved, paired with the good results achieved to date, are gradually moving this initiative to a more general purpose use, disconnected by the aims of the original ARIADNE project. The platform (for this reason recently renamed just to “Visual Media Service”), is also going towards a transformation process, consisting in enhancing the service possibilities and in extending it to other media types, following the same approach.

The results of this experience has been presented at the "CAA 2015 Conference on Computer Applications and Quantitative Methods in Archaeology" and published here:

- *ARIADNE Visual Media Service: Easy Web Publishing of Advanced Visual Media*, Ponchio F., Potenziani M., Dellepiane M., Callieri M. and Scopigno R., International Conference on Computer Applications and Quantitative Methods in Archaeology (CAA 2015), Volume 1, Chapter 6, page 433-442, March 2016; [1]

The current version of the Visual Media Service service can be accessed at the project webpage [2].

5.2 Edition Topoi Repository: Automating Web-Based 3D Publishing

The integration on the webpage of a single 3D model, or of a predefined restricted set of models, raises different issues compared to an effective integration of thousands of them in an online repository.

In the latter case, it becomes mandatory to have an automatized (or semi-automatized), pipeline aimed at homogenizing the dataset and driving the composition of the final Web presentation. This pipeline should be able to automatically manage and digest 3D data in all conditions, and display every single model with the best scene setup without any (or with a minimal) interaction by external actors.

Starting from these premises, in this section we will retrace the steps of a real application case-study [Edi16, PFDS16] aimed at publishing a large and heterogeneous three-dimensional dataset in a specialized Web repository.

In this experience, our goal was to introduce a viable and reusable strategy that, starting from the description of all the steps needed for properly pre-processing the data, would lead up to the implementation of the Web-based 3D viewer valid for all of them.

The resulting service-oriented publishing action exploited 3DHOP as core component of the pipeline, once again demonstrating its flexibility. This time our framework has been customized to adapt it to the several possibilities proposed by an heterogeneous and automatized scenario.

[1] © 2018 CAA. Text and figures reproduced by kind permission of CAA.

[2] `http://visual.ariadne-infrastructure.eu`

5.2.1 Overview

In the last decade there has been a proliferation of every type of online portal, platform, database and repository. The proposed solutions are very heterogeneous, addressed to different expertises (from humanities to science), developed following opposite philosophies (from restricted to Open Access), and with heterogeneous contents (from text to videos).

All these differences heavily affect the design choices to build the informative structure containing the published data, so as to make them accessible with the best user experience. From this specific point of view, the type of information people want to share plays a fundamental role, because of the inherently different nature of the various media (text, images, audio, and video) and also because of the different tools available to handle them (especially in a particular environment such as the Web).

Among the protagonists of this renewed publishing trend, 3D data deserve a special mention. Despite the necessity and utility of sharing 3D data online being clear, as we have already seen in Chapter 2 the wide number of available Web3D solutions seems to not be able to cover all the publishing needs. For instance, so far, nearly no effort has been aimed at facing the issues arising when the number of models and the level of automation required by the publishing process increase. Following these considerations, we decided to experiment with the idea of a service-oriented and semi-automatized publishing action, addressed to the Web integration of a large dataset of heterogeneous 3D models.

Exploring the specific context of the Open Access archaeological multimedia database (in the following presented), we will introduce a two step pipeline that, firstly defining a rough 3D data pre-processing, and then developing a "general purpose" Web viewer (i.e. able to visualize and analyze a large range of 3D model characterized by different basic technical features), will finally lead to obtain the specialized Web3D interactive visualization.

The Repository

The *Edition Topoi* [Edi16] is an innovative digital publication platform which holds, besides books and articles, a number of digital repositories for research data. Launched in April 2016, it is part of the interdisciplinary research project promoted by the *Topoi Excellence Cluster* [Top05] in Berlin, which combines different kinds of studies about ancient cultures. The publishing platform is strictly Open Access and the digital collections (see Figure 5.7) aim to be a long-term archive for the research on ancient cultures. All digital resources are considered as independent publications (called "citables"), thus can be cited in scientific publications and are statically reachable through the use of DOIs.

At the date of its release, the Edition Topoi Digital Library could count on eight repositories containing 3D data.

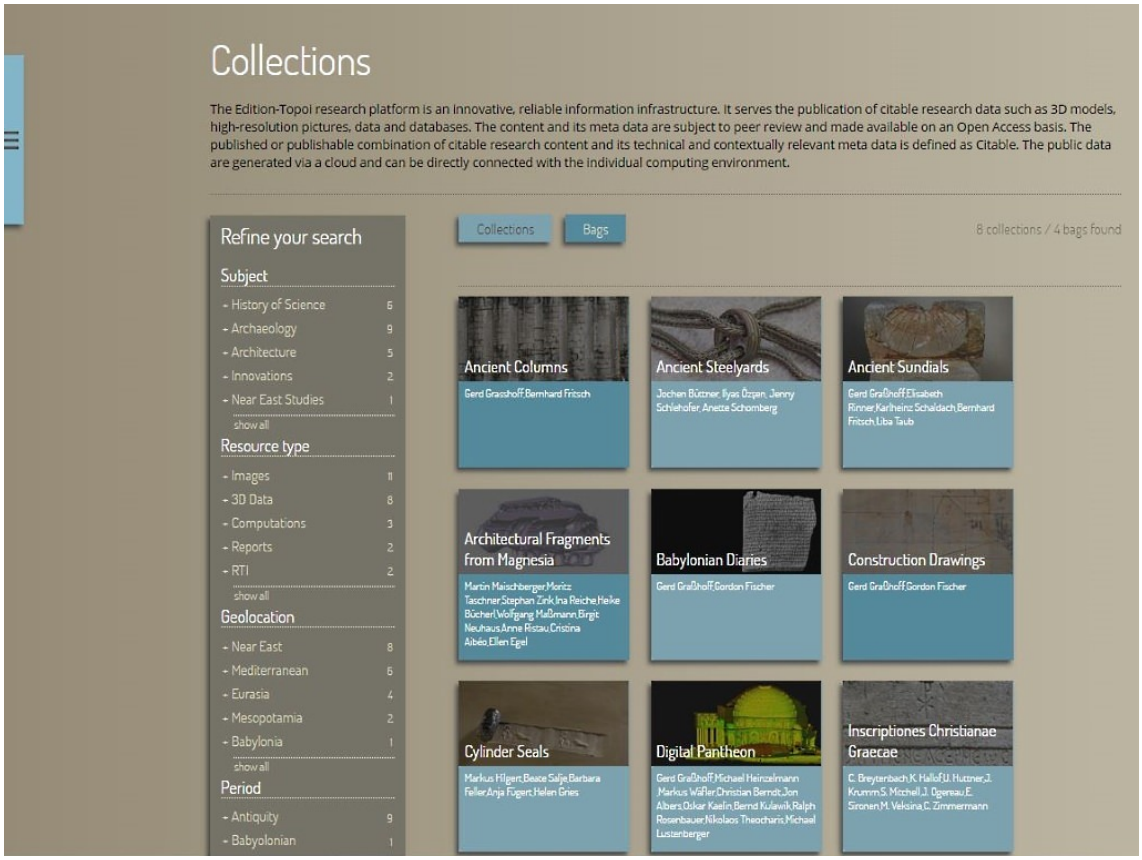

Figure 5.7: The picture shows the landing webpage of the Edition Topoi repository [Edi16], with all the various items composing the online database ordered in the established reference categories.

Acquired adopting different techniques, like Laser scanning, Structure-Light scanning or Structure-from-Motion (SfM), they result in 3D models of different size and types (point clouds or meshes, with or without colors, normal vectors, or other attributes, etc.). Moreover, since the Topoi association covers quite a number of different subjects (like Philosophy, Epigraphy, Archaeology or History of Science), and the data have been acquired in different projects over a longer period and in many different places (all over Middle Europe, the Mediterranean Sea and the Near East), the final dataset result to be very heterogeneous.

A short description of just some of the collections involved may clarify the need for a robust and reliable publication strategy:

- the Digital Pantheon Project, which is the only complete laser scan model of the Pantheon in Rome. The model was split into over 200 separated point clouds. They help in handling different research questions, like for example that one concerning the shape of the columns of the *portico*[GB11] (for that reason the visualization has to pay attention to the correct display of the

points);

- a collection of cylinder seals of the *Vorderasiatisches Museum - Staatliche Museen zu Berlin*. The over 1.200 objects of an average size from 2-3 cm resulted in point clouds about 200.000 and 900.000 vertices. The scans were used to create digital impressions of the seals where now different figures can be carved out. The technique of the fabrication of the seals can be analyzed as well;

- a collection of ancient sundials, which covers a wider range of 3D models. About 60 percent of the approximately 230 3D models were acquired with a Structure-Light scanner, the other 40 percent were obtained through the method of Structure-from-Motion. Both techniques provide meshes; the SfM models do additionally have color information. Since the objects are partly fragmented and partly quite large, the number of vertices and faces exhibits a lot of differences;

- a large number of architectural fragments of Magnesia on the Meander which are hold by the *Antikensammlung - Staatliche Museen zu Berlin*. The outcome of the project available in the repository include models of two different scanner techniques (Structure-Light and SfM).

All in all, the Edition Topoi Collections cover a broad spectrum and a large number (over 2.000) of 3D models based on the different subjects that have different research questions to digital data and the different methods that were used over the last ten years to create the models.

5.2.2 Publishing Service Deployment

The publishing of a full dataset of objects in a repository requires an elevated level of automation. It's clear that when the number of elements reaches the order of magnitude described in §5.2.1, a routine which contemplates to work with the single object is unfeasible, so what is really needed is a strategy that attacks the problem in its generality.

When the data to treat are three-dimensional (possibly big complexity, huge disk space usage, poor standardization and elevated intrinsic heterogeneity) and the database to implement is a Web-based database (limited computing resources, unknown available bandwidth, expensive server disk space), the issues increase significantly.

3D data indeed, frequently need to be processed to be prepared for a ready-to-use visualization, and these modifications generally require onerous computational actions, that can't be delegated to the network infrastructure, especially whereas they should be multiplied by the thousand instances of the repository objects.

Hence, it's mandatory to employ a working strategy able to move all the complex calculus on the dataset in a pre-processing stage (as automated as possible, given the amount of data involved) to execute locally. Only after this step, aimed to smooth the coarse differences between the input data, it will be possible to feed the database with the new output data, still considerably heterogeneous between them, but ready to be visualized in a viewer able to adapt itself to these differences.

So, in the following we present a pipeline (Figure 5.8) composed by two distinct working stages: the first one deputed to dataset pre-processing and the second one addressed to the Web viewer implementation.

The complete pipeline is strongly based on the 3DHOP framework, starting from the choice of the data format to exploit, up to the design of the final Web viewer.

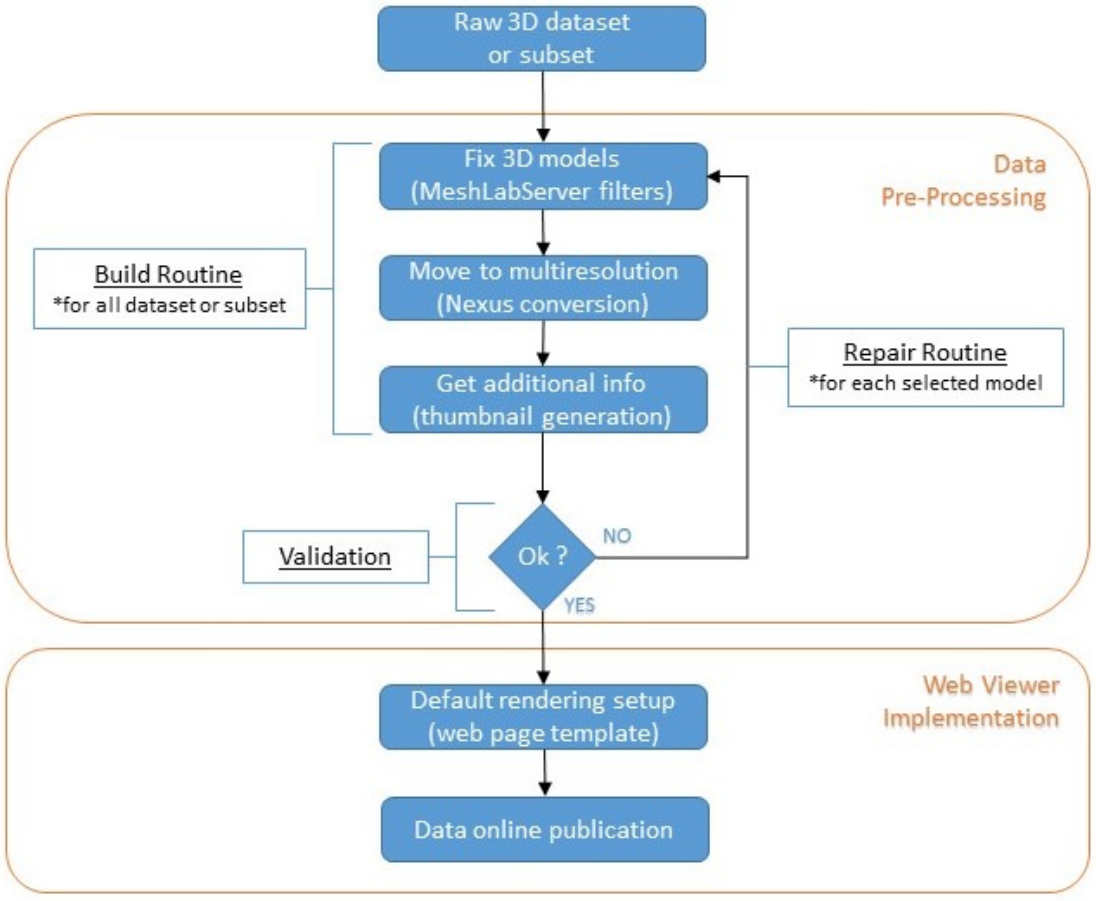

Figure 5.8: The schematic flowchart of the adopted pipeline, basically composed by two blocks: a first one deputed to 3D dataset pre-processing and the latter to the Web viewer configuration.

Dataset Pre-Processing

Several considerations have to be taken into account dealing with 3D models. Among them, data format and navigation are critical.

Concerning the first point, a big issue is related to the large number of formats available. Indeed, despite some attempts to define a common 3D format, a myriad of alternatives (PLY, STL, OBJ, OFF, PTX, VMI, DAE, PTS, XYZ, GTS, PDB, TRI, ASC, X3D, and WRL, just to name a few) is still around, and people working on 3D data are often forced to conversions among them. For instance, working with 3D dataset online is nearly mandatory to use multi-resolution data structures (to provide a seamless navigation in a short time), so data owners are usually obliged to move away from the classic single resolution 3D formats (widely used offline, but useful on the Web only when a limited amount of data has to be visualized). Even though nowadays several conversion tools are available, it is necessary to use them carefully, trying to preserve the attributes of 3D elements, since sometimes the limited standardization of the formats can lead to errors and loss of data. Also the definition of the interaction paradigm associated to a 3D model can be a big issue, especially in handling large and heterogeneous dataset, since the solutions adopted for instance to visit from inside a building reconstruction are almost certainly inappropriate to appreciate a digitization of a piece of ancient pottery, and *viceversa*.

These premises suggest that a pre-processing step may be needed to "prepare" the 3D models, and that this step could need some validation. Hence, an analysis of the input 3D data may help in creating the pre-processing stage.

The repository chosen for this test case can help to reduce the working area. First of all, all the models were associated to a similar origin (CH related) and to a similar intended use (scientific study or preservation). These simple constraints are fundamentals, because they cut off all the huge world of the modeled 3D data, where data format and topology could devise for different solutions in the processing stage.

Additionally, another important assumption concerns the navigation paradigm of the 3D scene: since the great majority of the models are archaeological findings digitized as single separated objects, all them well suite to a "turntable trackball" paradigm, where the object is treated as it is on the hand of the user, who is able to rotate the view around the object, and zoom in and out. The trackball paradigm can be defined using a few parameters that can be easily pre-calculated by analyzing the bounding box of the object. Other more accurate [Mal13] or *ad-hoc* [CDS15] approaches may be hard to extend to more complex surfaces.

For the trackball paradigm, the only important factor is that the model must be "aligned" to the axes of the reference system, otherwise the navigation is not realistic.

In order to take into account the needs and assumptions regarding the collections, we designed a smart batch pre-processing stage able to work on all the original data,

implemented as semi-assisted routine, able to provide a graphical feedback to the database developer and also to be programmable depending on the dataset needs.

This procedure (Figure 5.8, top), applying mathematical transformation on the entire 3D dataset, could easily need a lot of computational resource, so it needs to run locally, preferably on the same machine where the data are hosted.

The batch procedure is expected to:

- Fix some basic issues related to the models acquisition/generation (to do this we used the MeshLabServer service, a powerful highly programmable scripting tool provided by MeshLab [Vis05]);
- "Prepare" the models for the online environment, converting them from the original data format to the Web performing format handled by 3DHOP (to do this we exploit the Nexus package);
- (Optional) extract from the models some useful information to link in the database enriching the informative content of the related webpage (this can be obtained with several software/tools, depending on which information is needed).

More in detail, what we have done with the specific 3D data in our repository has been: the correction of the objects spatial positioning (since their alignment with the reference system was different w.r.t. the one used by the adopted Web viewer), the re-calculation of the normal vectors orientation, and the export in a 3D format (PLY) suitable to be used as input for the subsequent conversion.

Subsequently, the PLY files were converted in a the Nexus multiresolution compressed format. Then, a rendering of the 3D model from the initial point of view (given a basic turntable trackball paradigm) is generated. The image is intended to be used as a thumbnail, and as an output for user validation (see later).

These actions have been implemented in a script file, called "*build*". In detail, running the "*build*" batch procedure the system:

1. enters in the dataset location and visits all the directories and the sub-directories looking for the input model files;
2. for each 3D model found: it applies the MeshLabServer scripts with two selected filter (rotation and normal vectors calculation), saving the obtained fixed model in PLY format;
3. it feeds the Nexus converter with the obtained PLY files, getting as output the corresponding multi-resolution compressed Nexus model files;
4. for each final Nexus files: it opens it locally and take a screen-shot of the model saving a copy of it in an dedicated folder (called "*build*") unique for the entire database.

At the end of the first phase, in addition to the original model, and the generated PLY and Nexus models, there will be a "*build*" folder containing the thumbnails of every single object of the dataset.

The thumbnails are used by those who operate on data to understand which models need to be further processed (Figure 5.9).

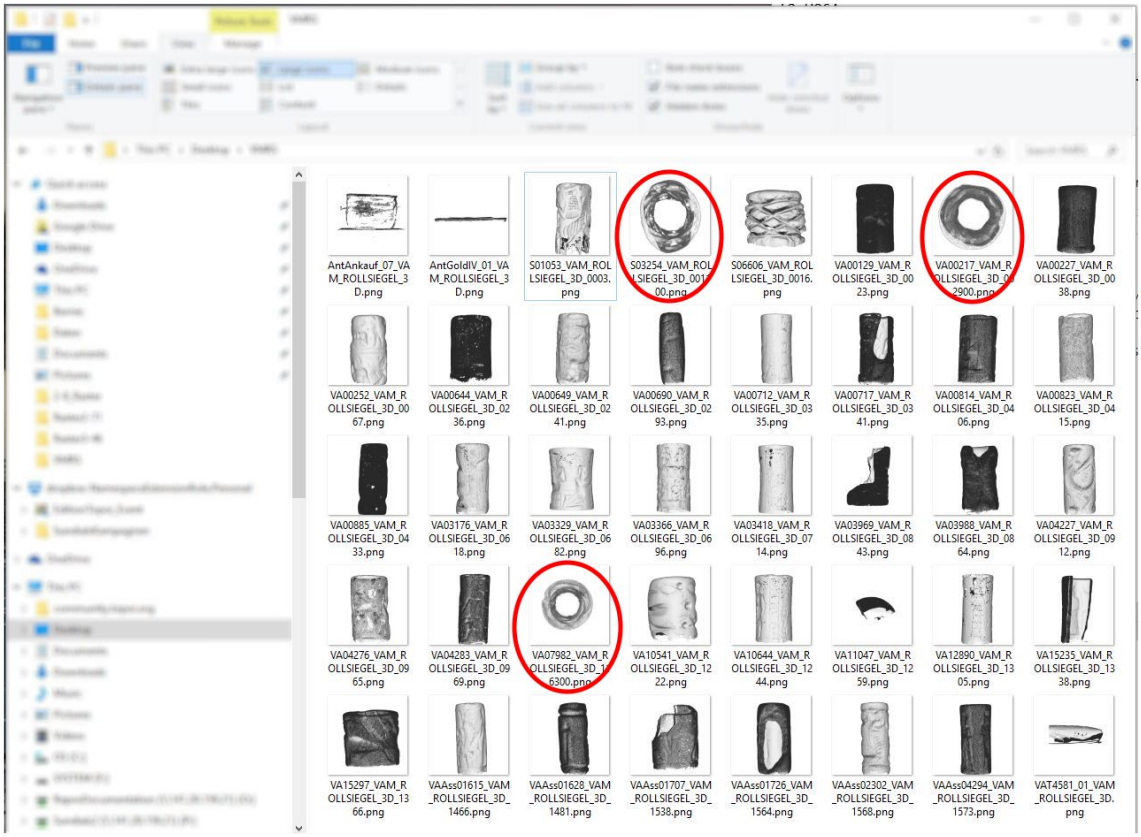

Figure 5.9: A figure showing a sample "*build*" folder with the thumbnails of 3D models after the first stage of the pre-processing pipeline. Examining the pictures of the models (almost all cylinder seals, in this case) it's easy to see which of them after the first rotation are still in an wrong initial position (thumbnails circled in red).

The second stage of the pre-processing is again implemented as a script. If the data operator finds a model that needs repairing (i.e. a further alignment in the reference space is needed), he moves the associated thumbnail from the "*build*" to another dedicated folder (called "*repair*"), then he selects the right MeshLabServer filter to apply (i.e. "*rotation*"), and finally he runs the "*repair*" batch script.

In the case of the collections taken into account, the MeshLabServer filters that could be applied were dedicated to two possible actions: rotation, and normal estimation. These operations could be needed to fix the right initial position of the model or have better estimated normals. The "*repair*" script, acting only on the repository models with the same name of the the thumbnails placed in the

homonyms folder, follows the same steps as the "*build*" one, finally updating with the new thumbnails the "*build*" folder where the user can check and validate the results again.

This pipeline structure, composed by sequential blocks, is highly configurable. The only mandatory constrains indeed are the kind of output format from the MeshLabServer stage (i.e. PLY) and the final 3D model file format (i.e. Nexus) which are fundamental for operating in an efficient way in the 3DHOP viewer.

All the other actions on the data can be decided by the data operator depending on the 3D dataset, just changing the associated pipeline block.

Designing and Developing the Web Viewer

The pre-processing pipeline aforementioned is fundamental to smooth the principal rough differences in the original dataset. However the previous step, while preserving the basic and fine features of the data (geometric structure, color information, etc), still preserves their differences and peculiarities.

Hence, it is important to choose an appropriate Web viewer, which should be able to:

- guarantee a solid and fast data streaming in "all terrains" conditions, supporting large amount of 3D data;
- automatically get (for every single model) the best scene setup, minimizing the intervention by the platform developer;
- support quite wide range of data setup possibilities (point clouds and meshes, textured model, with per vertex color or without any color, etc);
- allow customization according to the needs of the developer (so as to use it, for instance, in an automatized pipeline where the HTML element containing the 3D scene have to be generated with an unsupervised routine).

The 3DHOP framework satisfies all these requirements, virtually supporting all the data configurations of the repository dataset, and natively providing a set of tools addressed to the CH field. We just modified the basic implementation of the viewer in a couple of features, adding more flexibility and adaptability as required by a project like this, where the creation of the webpage cannot be supervised step by step.

The key point of this implementation (Figure 5.8, bottom) has been to act on the default settings of the scene, moving them from an hard coded choice to an exposed and programmable feature, giving more decision-making power both to the repository developer (who can set his preferred default values in the viewer template passed to the webpage) and to the final user (who can change the default setup interacting with the dedicated control panel shown in Figure 5.10 bottom left).

The source code of the viewer was edited, both at Shaders and JavaScript level, adding portions of code able to switch between different rendering modes and modifying the preexisting data structures. Appropriate informative flags, added to the JavaScript objects related to each model, have been used to describe the different structural characteristics of the 3D instances. The introduction of these flags gives the possibility to manage a couple of characteristics which are closely related with the virtual scene basic appearance: the geometric and the color representation of the 3D models. This basically allowed us to overcome the need to set the best scene configuration model by model.

The setup panel addressed to drive the rendering features (virtually supporting all the data configuration of the repository dataset) at the end has been integrate to the set of tools addressed to the CH field natively provided by 3DHOP. The final interface of the resulting Web3D viewer, aside from the basic navigation buttons (home and zoom controls), is so composed by a set of 4 different tool, in the following shortly described.

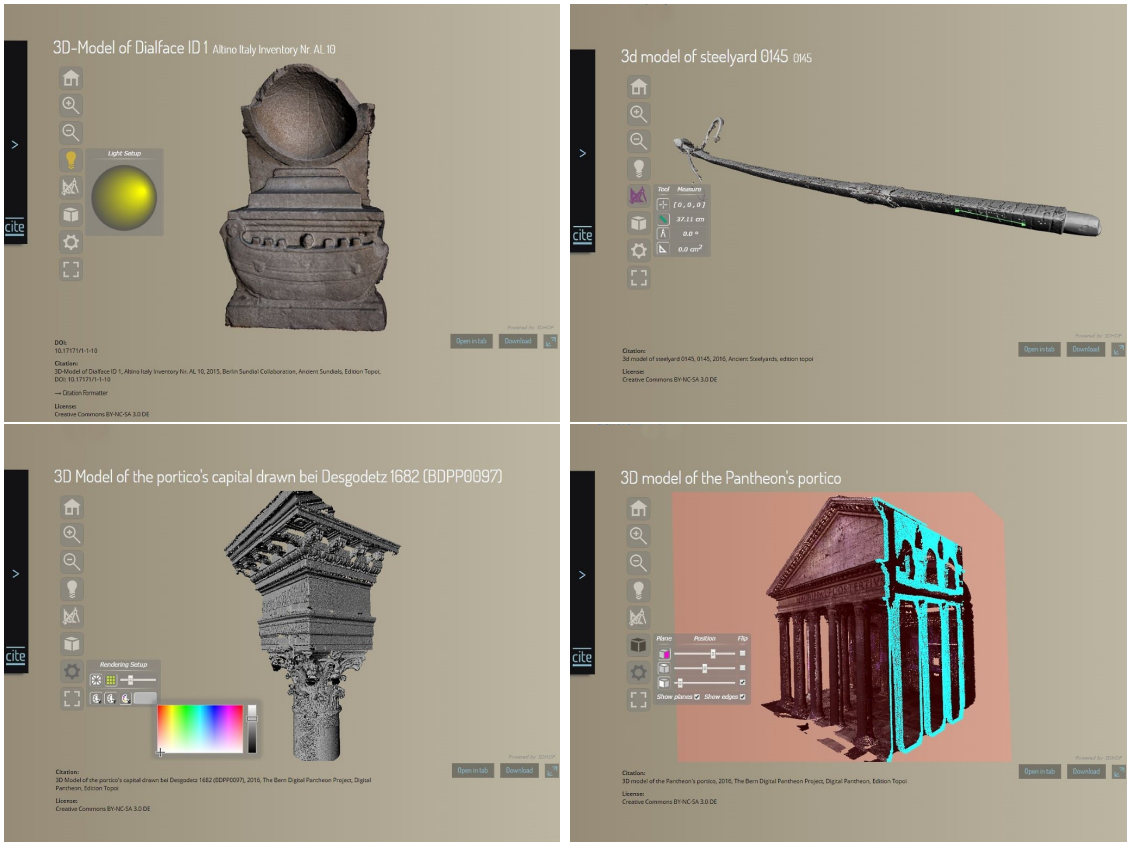

Figure 5.10: Four examples of tools implemented in the Edition Topoi [Edi16] viewer (in clockwise order starting from the upper left): light tool, measurement tool, sections tool, and setup tool.

The lighting control tool enables the control of the light source on the virtual scene: through a dedicated panel the user can interact with an intuitive HTML canvas (Figure 5.10 top left), moving and positioning the light source everywhere around the 3D scene. In the CH domain, directional light is a powerful mean to visualize fine geometric details.

The measurement button gives access to several tools (Figure 5.10 top right). By clicking on the measurement panel icons the repository user can select different interactive measurement possibilities, like for instance the point picker (for the retrieval of the 3D scene space coordinates of a point on the virtual scene), or the distance tool (for the calculation of the linear distance between two points), or again the angle tool (for the measurement of the plain angle between two lines).

Using the sectioning tool is possible to get planar sections of the model on the scene (Figure 5.10 bottom right). Three orthogonal sectioning planes are provided, the proper use of them not only give accurate references points (for measurement or other needs, for instance), but also allows to visualize the internal parts of the 3D objects. Interacting with the sliders in the section tool panel the final user can independently move each orthogonal section around the virtual scene; in addiction is possible also to control the section edges and planes visibility.

Finally, the already mentioned setup tool allows the user to modify in real time the scene rendering parameters. It gives to the final user the possibility to change the displayed primitives (switching between triangles and points), or to move from a textured to a color per vertex 3D model representation. It also allows to modify the size of the points in a point cloud rendering and, at last, to select a desired solid color for visualization (Figure 5.10 bottom left).

Of course, this implementation of the 3D Web viewer interface is just one among all the possibilities. The tools to add or to remove can be easily chosen by the database developer, who can freely select, for instance, to delete the sectioning tool, to separate the color setup panel from the rendering primitive setup panel, or even to integrate some hot-spot features to link external information directly inside the virtual scene.

5.2.3 Results

The strategy discussed in this case-study aims to drive, automate, and speed up, the publishing of a large set of 3D data in a web repository. In order to achieve these goals we proposed a two stage work flow, where a first (supervised) pre-processing of the entire raw dataset is performed, and then an online virtual scene setup is automatically generated.

The sheer size of the repository and its heterogeneous content have required a careful setup of an automated back-end, able to support the different pre-existing situations, and possibly to allow for a simpler management of future additions. All the work done in this part has been strictly connected with the idea of Web presentation, as it served also to define exactly what can be presented to the public,

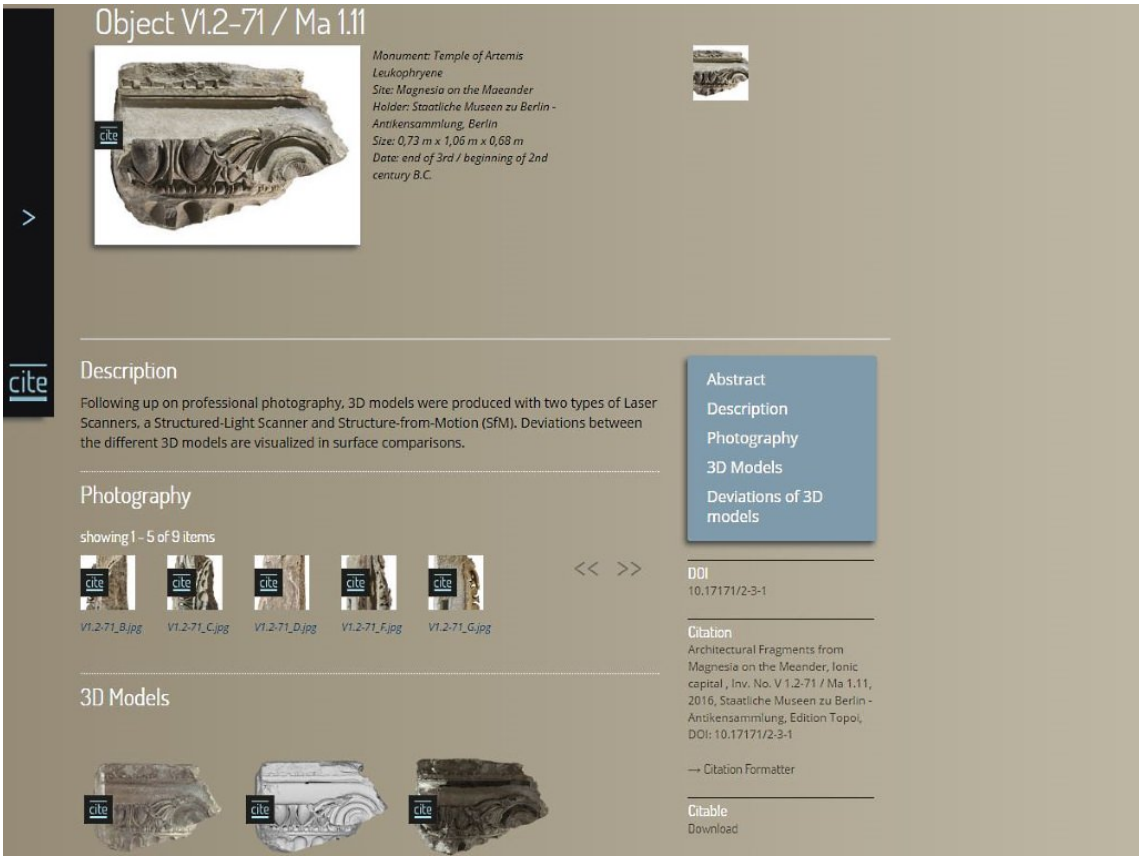

Figure 5.11: An example of webpage referring an object currently integrated in the Edition Topoi repository [Edi16].

and define a uniform, coherent access to the repository. We have seem that defining a specialized viewer for each one of the entities is impractical (if not time-impossible), but it is viable to start with a uniform solution for all the data, and then think, maybe on a "class" basis, to specialize the viewer to better work with that specific class of data. Completely custom viewers may then be created for one-of-a-kind objects and masterpieces. Thus, we have outlined a working scheme that rely on a first pre-processing phase (described in detail here) and then uses three levels of publications:

- a **basic generic viewer** available for all the entities in the repository, to provide easy access and preview, uniform and coherent browsing experience. This is the first step, done once to provide access to all the entities in the repository. It require a careful design of the basic "universal" viewer, as explained in the previous section, but it is still viable ("one viewer fits all"), and grants the most immediate and wide result.
- a **class-specific viewer** to exploit individual specificity of a class of entities (e.g. for all the sundial, add a "show time" visualization, where it is possible

to change the time of the day and see the shadow of the gnomon). This is the second step, and may be seen as an incremental result, as we are specializing the existing viewer, thus only requiring a moderate, focused effort.

- a **fully custom viewer** for few, selected objects, to be used as showcase of masterpieces and peculiar object. This is the final step, and can be done when resources are available, or on a per-needed base, again, optimizing the design and development effort.

It is our belief that this working strategy is efficient and sustainable, as it optimize the effort for developing and deploy the repository, and fits perfectly the 3DHOP workflow, as the incremental specialization of the viewer is possible due to the modularity of our system, and does not require a complete rewrite of the viewer code, still preserving a strong uniformity in terms of interaction and rendering. Future addition to the repository will follow the same workflow: after the preprocessing step they will be homogenized to the rest of the repository, and will receive the generic viewer (or the generic class viewer, if available), resulting in an immediate availability.

The pre-processing of the full 3D dataset, carried up on a common machine with a few batch procedures, has allowed to prepare the data to be efficiently handled online. Using the output of this first working stage, the second stage of the proposed work flow has been performed without complications, exploiting the 3DHOP capabilities and possibilities. The generation of the webpages related to each single object has been obtained with standard Web routines used in the development of similar online structure (i.e. using templates) with the only addition of the design pattern able to produce the 3D viewer implementation.

Thanks to this pipeline, the entire Web repository has been populated in a short time with the 3D models introduced in 5.2.1. The resulting Edition Topoi digital library has been released in April 2016, and it is currently accessible online [Edi16].

In the current implementation, the Web repository allows the access to 18 different classes of items, composed by a mixed set of data: images, reports, RTIs, computations and also 3D models (present in 12 of the mentioned classes).

For each class, the user gets information about the selected subset objects. Subsequently he can refine the search until landing on the page of a single object. Here all the informative multimedia layers (text, pictures, 3D, RTIs, etc) are shown, organized in a rational and sensible way (Figure 5.11).

The thumbnail of the 3D object (coming from the pre-processing stage) gives access to the Web viewer page, where besides to other textual information related to the model, there is the pre-defined virtual scene deployed with 3DHOP.

3DHOP played a central role in this experience, influencing both the pipeline stages. This case-study resulted very important in the 3DHOP validation, proving once more the versatility and the strengths of our framework, also when adapted to application context slightly different from a common publishing action.

The results of this experience has been presented at the "STAG 2016 Conference on Smart Tools and Apps for Graphics" and published here:

- *Automating Large 3D Dataset Publication in a Web-Based Multimedia Repository*, Potenziani M., Fritsch B., Dellepiane M. and Scopigno R., Eurographics Conference on Smart Tools and Apps for Graphics (STAG 2016), page 99-107, October 2016; [3]

The final Web3D repository of the *Topoi Excellence Cluster* can be accessed at the project webpage [4].

[3]

[4] `http://repository.edition-topoi.org`

Chapter 6

Conclusions

With this thesis, we aimed at exploring the Web3D environment, trying to provide a sensible answer to the need for effective approaches to 3D Web publishing. Nowadays, the Web integration of 3D content is a very dynamic and rapidly-evolving world, populated by a myriad of different technical solutions. Despite the great variety of possibilities, at a closer analysis the resulting scenario is characterized by unsolved issues, uncovered users, and neglected fields. The presence of these "missing links" has been the main motivation of this work.
Starting from these premises, we decided to investigate the Web3D landscape, studying its evolution and identifying current trends, with the aim to define the shortcomings and deficiencies of current approaches/technologies, and to clearly identify open challenges. To conduct this state-of-the-art and characterization review we considered both academic literature (selecting about a hundred remarkable works) and software systems (again, selecting about a hundred of significant cases). The panorama resulting from this study is definitely heterogeneous and complex. To produce a more synthetic view we decided to organize our work primarily analyzing in detail the possibilities provided by the major current approaches, secondly isolating some of the prominent and recurring features and capabilities offered by these systems, then grouping them by their scope and functionality, and finally cross-mapping those characteristics with the requirements of different application domains.
This work, reported in detail in Chapter 2, has been submitted recently as a monograph to the journal "Foundations and Trends in Computer Graphics and Vision" and is currently under review.

The main results of this thesis is the design of a new platform for the production of interactive Web3D content, called *3D Heritage Online Presenter (3DHOP)* [PCD+15a]. This platform derives from the characterization presented in Chapter 2 and is aimed at solving the open issues individuated in that review work. The goal here was to contribute to the requirements and open issues of a specific application domain, Cultural Heritage, even though the resulting platform

is not strictly limited to that domain. 3DHOP has been designed to enable the fast and easy publication of 3D models online, supporting users both with the required technology and with the didactic resources to train and introduce them using it. Other key objectives in the design have been to provide a flexible, modular and performing platform, fully integrated in the Web ecosystem.
The 3DHOP solution is a Web-based framework aimed at the interactive visualization of complex 3D dataset (Chapter 3). Its final implementation has given a strong emphasis on some key aspects, e.g. efficient formats/approaches for 3D data transmission/rendering, intuitive interaction/manipulation paradigms, and features enabling the connection of trans-media elements, globally aiming at simplifying the creation of Web pages able to interactively display high-resolution 3D models.
Initially conceived as a demonstrator, and then evolved in a real system, now 3DHOP can be considered a fully qualified solution with a consolidated community of users. It is distributed as an open source tool (which includes also user guide resources, explaining step by step how to use and configure the different layouts and components provided). To date the 3DHOP package counts around 80 downloads per month, while the 3DHOP official website [Vis14a] is visited around 800 times per month. 3DHOP is currently used by tens of different academic, educational and commercial entities, which have developed their own visualization webpages (in most cases without our support or help).
The value of the 3DHOP platform was recognized by the DH community, since it got the honorable mention (second place) in the category "Best DH Tool or Suite of Tools" of the *Digital Humanities Awards 2015* [1].

Over the time we had highly encouraged the extension and modification of our framework, gathering suggestions and feedback from the users. These third parties utilizations and extensions of 3DHOP (some representative examples are presented in Figure 3.8) have been useful also to operate a preliminary validation on our platform. Indeed, despite in most cases those external users exploited just the basic layout/configuration or only proposed some basic customizations, these independent testbeds can be already seen as a sufficient demonstration of the 3DHOP usability/adaptability in different application scenarios, and consequently, as a validation of our design choices.
However, to have a better understanding on the real effectiveness of our solution we have also contributed to the implementation of more comprehensive testbeds. In particular, we presented in the thesis two couples of real application examples addressing different needs: the first two test cases providing examples of 3DHOP used as specialized content creation tool, and the second two providing examples of 3DHOP used as a basic resource for the implementation service-oriented applications.

[1] `http://dhawards.org/dhawards2015/results`

Concerning the specialized *Web3D content creation* (Chapter 4), we first presented a publishing experience specialized to supporting a content dissemination action (the Pollock case), and then a second one developed for supporting scientific analysis (the study of the polycromy over a Roman sarcophagus). These publishing actions, even if aimed to cover different needs, are both characterized by a data-centric design, that has required to strongly adapt the 3DHOP features to the specific aims of the two experiences, in this way allowing us to better validate the modularity and the flexibility of our framework.

More specifically, in the first contribution we presented a publishing experience that, starting from a 3D digitization campaign of painting by Pollock required to support conservative and diagnostic purposes, has finally led to the design of a Web3D application to be used both in a virtual exhibition (as a Web application) and in a real exhibition (as a museum kiosk).
In this case, through an important customization of 3DHOP, we developed a presentation tailored on the 3D geometry of the analyzed artwork, that gave to the visitors the possibility to interactively understand the most interesting aspects and the complexity of the artwork, somehow providing an unconventional experience. The resulting 3D application was part of the temporary exhibition *"ALCHEMY BY JACKSON POLLOCK. Discovering the Artist at Work"*, inaugurated in the Peggy Guggenheim Collection in Venice in 2015 and lasted eight months, totaling more than 180.000 visitors. The same kiosk was also on exposition in a similar initiative hosted by the Guggenheim Collection Museum in New York in 2016-2017. The Web version of the kiosk it is accessible online at [Vis15c].
The results of this experience, presented at the "IEEE Digital Heritage 2015 International Conference" [CPP+15], won the conference *Best Paper Award*. This project also won the first prize in the category "Best Use of DH For Public Engagement" at the *Digital Humanities Awards 2015* [2].

In the second contribution we presented a publishing action aimed at building a technical tool for supporting the study and the documentation of an archaeological artifact. In this case-study, focusing on a proper interface design, we built a specialized Web3D application able to effectively present the heterogeneous set of analytical examinations performed on an Roman painted sarcophagus. The resulting Web documentation exploits the digitized 3D geometry of the artifact as the main channel to manage and display, integrated in a single reference system, all the scientific analysis and historical information collected. Also in this case, the final result (accessible online at [Vis15d]), was tested in a real applicative context, and contributed to the validation of the effectiveness of the proposed solution.
The results of this experience was presented at the "IEEE Digital Heritage 2015

[2] `http://dhawards.org/dhawards2015/results`

International Conference" [SPPS15].

Concerning the application examples addressing the *service-oriented 3D publishing* (Chapter 5), we presented firstly an experience aimed at developing a media publishing service platform, and then a contribution aimed at supporting an automatized publishing pipeline for a Digital Library.
Indeed, both these case studies have been designed to exploit 3DHOP as low-level code base for providing higher-level solutions. This approach, requiring customizations that touch innermost layers of our framework, has allowed us to evaluate the 3DHOP adaptability from a more technical perspective.

The first contribution describes the steps needed in design and the develop of an online service aimed at the assisted publishing of complex data (3D content, 2D Hi-Res Image, and RTI). In this testbed 3DHOP has been used as code base for implementing a server based visualization platform, as well as pattern design for the structures addressed at managing the two not 3D visual media not natively handled by 3DHOP. This experience resulted in the ARIADNE Visual Media Service [Vis15b], a one-click service able to simply convert the media file uploaded into a performing Web visualization.
The results of this experience was presented at the "CAA 2015 Conference on Computer Applications and Quantitative Methods in Archaeology" [PPD+15].

The second testbed was aimed at supporting the semi-automatic processing and publishing of an extensive and heterogeneous set of 3D digitalizations, with the final goal to build an enriched multimedia Web repository (set up by the Topoi Excellence Cluster). In this case-study 3DHOP has been exploited for designing a pipeline (composed by a pre-processing plus a visualization stage) useful to rationalize and speedup the deployment of the specialized Web3D repository. The developed Web Digital Library (the Edition Topoi Collection [Edi16]) provides free access to thousand of 3D models effectively integrated with other media informations, endorsing their interactive analysis through the different tools composing the user interface.
The results of this experience was presented at the "EG STAG 2016 Conference on Smart Tools and Apps for Graphics" [PFDS16].

6.1 Future Works

3DHOP is a living platform, it has already reached a quite mature status (the current version released on the Web is already v.4.1) and reached very good figures for both external users, downloads and accesses to the webpage. We have extensively used the 3DHOP platform in a number of projects concerning CH (many more than the four examples presented in the thesis). Those application

cases allowed us to further assess its flexibility and extendibility.

Designing, developing, and testing a framework for 3D Web publishing has given us the chance to carefully study any single facet of Web3D. The knowledge acquired closely analyzing this world has returned us a clear picture of past and present Web3D, but has also provided us with a perspective about its future, still characterized by a rapidly evolving nature, open issues, and possibilities to explore. With these premises it is clear that the searching for effective approaches to 3D Web publishing cannot be considered a terminated subject.

A brief list of ideas and suggestions easily to become research themes for future works on Web3D may include studies on:

- **Innovative interaction paradigms**, exploring the specialized features that are required for enlarging the toolset of supported devices and for improving the Web3D application accessibility. Nowadays, the research spaces in this direction are mostly driven by the widespread diffusion of new AR/VR technologies and I/O devices, that are opening the door to new navigation possibilities still to explore in Web applications. But for Web3D applications *novel devices* also mean *mobile devices*, which actually opened to the wide and stable support of WebGL only recently. Unfortunately, beside an heterogeneous approach in enabling the Web 3D API features, mobile devices also show considerable differences in many other central components, like for instance the handling the Web touch events, or also the management of the CANVAS images drawing in the context of high pixel densities (DPI) displays. All that, together with the lack of pattern design universally recognized, makes particularly challenging to develop Web3D solutions for these devices. So, it easy to foresee that the accessibility of Web3D applications on mobile or innovative platforms will be a breeding research ground in the near future, and, for this reason, it certainly will steer our plans for further works on 3D Web publishing.

- **Improved features for scene creation**, investigating novel and more flexible modalities for defining a 3D scene. This development area concerns the investigation of additional initialization features/schemes, but also the searching for easier definition paradigms. We know that, in order to be fully compliant with the Web environment, Web3D solutions will have to improve this area, maybe borrowing or hybridizing features from Web-based platform/services. Currently, authoring helpers and wizard tools seem to be the only way to solve this outstanding issue. But at the moment few solutions adopt an effective visual editor able to provide, at the same time, simplicity of use and a fully set up the visualization scheme. Conversely, the ideal Web3D authoring tool (beside to accept the upload of 3D models and

their metadata), should provide sensible defaults for the scene setup, without preventing the content creator to easily customize any 3D scene features just using simple Web forms/interfaces. Since the scene creation dichotomy represents a severe hurdle in approaching Web3D, we think that in our future works on 3D Web publishing may be valuable to investigate server-based guided tools specifically developed for 3D, and, most importantly, to propose similar solutions also for client-based platforms.

- **Specialized trans-media mapping**, concerning more sophisticated features for integrating different (peculiar) data types on the Web. A practical example could be the case where we have several images conveying different information that have to be mapped on a 3D object. Web3D solutions already enable textured models, but for specific applications would be valuable supporting the switching of multiple textures active on the same mesh (e.g. RGB data, UV images, multi-spectral textures, or any image-based scientific investigation produced on the same surface), eventually handling distortion issues coming from the use of different specialized acquisition devices. But, at the same time, trans-media mapping can be seen from the opposite point of view, i.e. such as the chance to map 3D on other media. A practical example of that could be the integration of 3D models into panoramic images, to enable an enriched visualization of these 360° environments. This addition could provide a more natural way of interfacing with the spatial features, enabling users to identify, query or analyze the 3D information integrated in the panoramic images environment. In the light of this it easy to imagine that sophisticated trans-media mapping will be a central feature to explore in our next studies on Web3D publishing.

- **Advanced management of the information**, moving towards shared documentation platforms, and studying new modalities to attach or link informative layers to the 3D model. Current solutions include the *annotation* and *hot spot* concepts, which are a first attempt to link any Web-enabled data (an image, some text, etc.) to a geometric location in the 3D scene. Nevertheless, there are several applications requiring more sophisticated approaches, where is needed a flexible support for the effective definition of regions (which could be points, polylines, or polygonal regions) where to attach additional data. Moreover, could be also useful to enable final users to contribute to these additional information, through a review/editing interface requiring (complex) real-time routines. In all those cases, future approaches will have to provide data organization and archival features, rather than simply visualization. Therefore, it is likely to think that our future works on Web3D will be aimed at effectively integrating data creation/ingestion to the underlying database, to store in a structured way all the handled information.

6.2 Thesis Achievements

Papers published (in chronological order):

- *Alchemy in 3D - A Digitization for a Journey Through Matter*, Callieri M., Pingi P., Potenziani M., Dellepiane M., Pavoni G., Lureau A. and Scopigno R., IEEE Digital Heritage 2015 (DH 2015) International Conference, Volume 1, page 223-231, October 2015; [CPP+15]

- *Digital Study and Web-based Documentation of the Colour and Gilding on Ancient Marble Artworks*, Siotto E., Palma G., Potenziani M. and Scopigno R., IEEE Digital Heritage 2015 (DH 2015) International Conference, Volume 1, page 239-246, October 2015; [SPPS15]

- *3DHOP: 3D Heritage Online Presenter*, Potenziani M., Callieri M., Dellepiane M., Corsini M., Ponchio F. and Scopigno R., Computers & Graphics (C&G) International Journal, Elsevier, ISSN: 0097-8493, Volume 52, page 129-141, November 2015; [PCD+15a]

- *3DHOP: una piattaforma flessibile per la pubblicazione e visualizzazione su Web dei risultati di digitalizzazioni 3D*, Potenziani M., Callieri M., Dellepiane M., Corsini M., Ponchio F. and Scopigno R., Archeomatica, ISSN: 2037-2485, Volume 6 (4), page 6-11, December 2015; [PCD+15b]

- *ARIADNE Visual Media Service: Easy Web Publishing of Advanced Visual Media*, Ponchio F., Potenziani M., Dellepiane M., Callieri M. and Scopigno R., Computer Applications and Quantitative Methods in Archaeology (CAA 2015) International Conference, Volume 1, Chapter 6, page 433-442, March 2016; [PPD+15]

- *Automating Large 3D Dataset Publication in a Web-Based Multimedia Repository*, Potenziani M., Fritsch B., Dellepiane M. and Scopigno R., Smart Tools and Apps for Graphics (STAG 2016) Eurographics Conference, page 99-107, October 2016; [PFDS16]

- *Delivering and using 3D models on the web: are we ready?*, Scopigno R., Callieri M., Dellepiane M., Ponchio F. and Potenziani M., Virtual Archaeology Review (VAR) International Journal, ISSN: 1989-9947, Volume 8 (17), page 1-9, July 2017; [SCD+17]

- *A Web-based system for data integration and visualization of the ancient colour*, Siotto E., Palma G., Potenziani M. and Scopigno R., 7th Round Table on Polychromy in Ancient Sculpture and Architecture (Florence 2015) International Conference, ed. P. Liverani, page 183-188, 2018 (in press); [SPPS18]

Submitted papers:

- *Publishing and Consuming 3D Content on the Web, a Survey*, Potenziani M., Callieri M., Dellepiane M. and Scopigno R., Foundations and Trends (in Computer Graphics and Vision) International Journal, pages 89, 2018 (submitted, under review); [PCDS18]

Awards achieved by results produced in this Ph.D.:

- The tool *3DHOP* [Vis14a] was 1st runner up in the category "Best DH Tool or Suite of Tools" at the **Digital Humanities Awards 2015**.
- The project *Alchemy in 3D* [Vis15c] won the first prize in the category "Best Use of DH For Public Engagement" at the **Digital Humanities Awards 2015**.
- The paper *Alchemy in 3D - A Digitization for a Journey Through Matter* [CPP+15] won the **Best Paper Award** at the "IEEE Digital Heritage 2015 International Conference".

6.3 Acknowledgments

This thesis has been the final outcome of a group effort, mostly resulting from the strong collaboration with the Visual Computing Lab[3]. The achievement of the goals presented in this work would not have been possible without the fundamental contribution of the people listed in the following:

- Chapter 3 – *Federico Ponchio* [3], in the development of the Nexus engine multi-resolution core;
- Chapter 4 (§4.1) – *Paolo Pingi* [3], in the "Alchemy in 3D" project management and in the data acquisition;
- Chapter 4 (§4.2) – *Federico Ponchio* [3], in the "Visual Media Service" project management and in the database deployment;
- Chapter 5 (§5.1) – *Eliana Siotto* [3], in the "Color and Gilding on Ancient Marbles" project management and in the data acquisition;
- Chapter 5 (§5.2) – *Bernhard Fritsch* [4], in the "Edition Topoi" project management and in the data gathering.

[3]Visual Computing Lab, ISTI CNR, Pisa, Italy
[4]Excellence Cluster Topoi, HU, Berlin, Germany

Bibliography

[3D 12a] 3D Technologies R&D. 3D Wayfinder. URL `https://3dwayfinder.com`, 2012. 17, 49

[3D 12b] 3D Technologies R&D. Frak Engine. URL `https://github.com/evanw/lightgl.js`, 2012. 17

[AD01] P. Alliez and M. Desbrun. Progressive compression for lossless transmission of triangle meshes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 195–202, New York, NY, USA, 2001. ACM. 15

[Ado11] Adobe. Stage 3D. URL `http://www.adobe.com/devnet/flashplayer/stage3d.html`, 2011. 10

[AEB13] J. Agenjo, A. Evans, and J. Blat. Webglstudio: A pipeline for webgl scene creation. In *Proceedings of the 18th International Conference on 3D Web Technology*, Web3D '13, pages 79–82, New York, NY, USA, 2013. ACM. 17

[AEB15] A. L. Ahire, A. Evans, and J. Blat. Animation on the web: a survey. In *Web3D*, 2015. 29

[Age13] J. Agenjo. WebGL Studio. URL `http://webglstudio.org`, 2013. 17, 26, 32

[Agi10] Agisoft. PhotoScan. URL `http://www.agisoft.com`, 2010. 99

[AH15] F. Anfinsen and K. Hope. ShareMy3D. URL `https://sharemy3d.com`, 2015. 25, 39, 44

[Amb10] Ambiera. CopperLicht. URL `http://www.ambiera.com/copperlicht`, 2010. 41, 44, 51

[Arc14] Archilogic. Archilogic. URL `https://spaces.archilogic.com`, 2014. 29, 33, 43, 49

[Aut98] Autodesk. Maya. URL `https://www.autodesk.com/products/maya`, 1998. 11, 32

[Aut11] Autodesk. Tinkercad. URL `https://www.tinkercad.com`, 2011. 23, 33, 50, 51

[Aut13] Autodesk. Fusion 360. URL `https://www.autodesk.com/products/fusion-360`, 2013. 42

[Aut15] Autodesk. Remake. URL `http://remake.autodesk.com`, 2015. 28, 40, 44, 45, 47, 48, 50, 51

[BCK+11] A. Blume, W. Chun, D. Kogan, V. Kokkevis, N. Weber, R. W. Petterson, and R. Zeiger. Google body: 3d human anatomy in the browser. In *ACM SIGGRAPH 2011 Talks*, SIGGRAPH '11, pages 19:1–19:1, New York, NY, USA, 2011. ACM. 22

[BCR05] J. Biström, A. Cogliati, and K. Rouhiainen. Post-wimp user interface model for 3d web applications. 2005. 35

[BD07] D. Brutzmann and L. Daly. *X3D: Extensible 3D Graphics for Web Authors*. Morgan Kaufmann, 2007. 8

[BEJZ09] J. Behr, P. Eschler, Y. Jung, and M. Zöllner. X3dom: A dom-based html5/x3d integration model. In *Proceedings of the 14th International Conference on 3D Web Technology*, Web3D '09, pages 127–135, New York, NY, USA, 2009. ACM. 13

[Bel11] N. Belmonte. PhiloGL. URL `http://www.senchalabs.org/philogl`, 2011. 11, 51

[BGB+05] L. Borgeat, G. Godin, F. Blais, P. Massicotte, and C. Lahanier. Gold: Interactive display of huge colored and textured models. *ACM Trans. Graph.*, 24(3):869–877, July 2005. 63

[BH10] M. Buckwald and D. Holz. Leap Motion. URL `https://www.leapmotion.com`, 2010. 36

[Bie87] E. A. Bier. Skitters and jacks: Interactive 3d positioning tools. In *Proceedings of the 1986 Workshop on Interactive 3D Graphics*, I3D '86, pages 183–196, New York, NY, USA, 1987. ACM. 32

[Bit02] Bitmanagement Software. BS Contact. URL `http://www.bitmanagement.com`, 2002. 13

[BJFS12] J. Behr, Y. Jung, T. Franke, and T. Sturm. Using images and explicit binary container for efficient and incremental delivery of declarative 3d scenes on the web. In *Proceedings of the 17th International Conference on 3D Web Technology*, Web3D '12, pages 17–25, New York, NY, USA, 2012. ACM. 57

[BJK+10] J. Behr, Y. Jung, J. Keil, T. Drevensek, M. Zoellner, P. Eschler, and D. Fellner. A scalable architecture for the html5/x3d integration model x3dom. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 185–194, New York, NY, USA, 2010. ACM. 13

[BKLP01] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev. An introduction to 3d user interface design. *Presence*, 10:96–108, 2001. 30

[BKLP04] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004. 30

[Ble95] Blender Foundation. Blender. URL `https://www.blender.org`, 1995. 11

[Bli04] Blizzard Entertainment. World of Warcraft. URL `https://worldofwarcraft.com`, 2004. 33

[Blo17] BlocksCAD. BlocksCAD. URL `https://www.blockscad3d.com`, 2017. 51

[Boe09] E. S. Boese. *An Introduction to Programming with Java Applets*. Jones & Bartlett Learning, 2009. 9

[Box16] Boxshot. Koru. URL `http://boxshot.com/koru`, 2016. 42, 50

[Bru10] P. Brunt. GLGE - WebGL for the lazy. URL `http://www.glge.org`, 2010. 11

[Buz15] A. Buzin. WhitestormJS. URL `https://whsjs.io`, 2015. 39

[Cab10] R. Cabello. Three.js. URL `http://threejs.org`, 2010. 13, 17, 36, 39, 58

[CCCS08] M. Callieri, P. Cignoni, M. Corsini, and R. Scopigno. Masked Photo Blending: Mapping Dense Photographic Data Set on High-resolution Sampled 3D Models. *Comput. Graph.*, 32:464–473, 2008. 100

[CDS15] M. Callieri, M. Dellepiane, and R. Scopigno. Remote visualization and navigation of 3d models of archeological sites. In *ISPRS Archives, Proceedings of 3D-ARCH Conference*, volume XL-5/W4, pages 147–154, March 2015. 124

[Ces11] Cesium Consortium. Cesium. URL `https://cesiumjs.org`, 2011. 43, 49

[CGG+05] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. Batched multi triangulation. In *Proceedings IEEE Visualization*, pages 207–214, Conference held in Minneapolis, MI, USA, October 2005. IEEE Computer Society Press. 15, 63

[CH15] X. Cheng-Hong. LayaAir. URL `http://www.layabox.com`, 2015. 41, 51

[Chu12] W. Chun. Webgl models: End-to-end. In Patrick Cozzi and Christophe Riccio, editors, *OpenGL Insights*, pages 431–454. A K Peters/CRC Press, 2012. 22

[CLDS13] M. Callieri, C. Leoni, M. Dellepiane, and R. Scopigno. Artworks narrating a story: a modular framework for the integrated presentation of three-dimensional and textual contents. In *Web3D, 18th International Conference on 3D Web Technology*, pages 167–175, June 2013. 59

[CMS88] M. Chen, S. J. Mountford, and A. Sellen. A study in interactive 3-d rotation using 2-d control devices. In *SIGGRAPH*, 1988. 31

[COJ15] J. Chandler, H. Obermaier, and K. I. Joy. WebGL-Enabled Remote Visualization of Smoothed Particle Hydrodynamics Simulations. In E. Bertini, J. Kennedy, and E. Puppo, editors, *Eurographics Conference on Visualization (EuroVis) - Short Papers*. The Eurographics Association, 2015. 22

[Con12] J. Congote. Medx3dom: Medx3d for x3dom. In *Web3D*, 2012. 21, 49

[CPP+15] M. Callieri, P. Pingi, M. Potenziani, M. Dellepiane, G. Pavoni, A. Lureau, and R. Scopigno. Alchemy in 3d: A digitization for a journey through matter. *2015 Digital Heritage*, 1:223–230, 2015. 84, 135, 139, 140

[CR13] D. Catuhe and D. Rousset. BabylonJS. URL `https://www.babylonjs.com`, 2013. 27, 39, 44, 51

[CSH+92] B. D. Conner, S. S. Snibbe, K. P. Herndon, D. C. Robbins, R. C. Zeleznik, and A. van Dam. Three-dimensional widgets. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, I3D '92, pages 183–188, New York, NY, USA, 1992. ACM. 32

[Cur00] D. H. Curtis. *Flash Web Design: The Art of Motion Graphics*. New Riders Publishing, Thousand Oaks, CA, USA, 2000. 8, 9

[DBGB12] M. Di Benedetto, F. Ganovelli, and F. Banterle. Features and design choices in spidergl. In Patrick Cozzi and Christophe Riccio, editors, *OpenGL Insights*, chapter 41, pages 583–604. A K Peters/CRC Press, 2012. 11

[DBPGS10] M. Di Benedetto, F. Ponchio, F. Ganovelli, and R. Scopigno. Spidergl: A javascript 3d graphics library for next-generation www. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 165–174, New York, NY, USA, 2010. ACM. 11

[DC08] S. Dumazet and P. Callet. Visual appearance of a middle age sculpture: Polychromy and lighting. In *14th International Conference on Virtual Systems and Multimedia*, pages 215–219, Cyprus, 2008. 100

[DC11] T. Da Costa. Lagoa. URL `http://home.lagoa.com`, 2011. 42, 43, 50

[DDF+10] F. Dupont, T. Duval, C. Fleury, J. Forest, V. Gouranton, P. Lando, T. Laurent, G. Lavoué, and A. Schmutz. Collaborative scientific visualization: The collaviz framework. 2010. 9

[DeL10] B. P. DeLillo. Webglu development library for webgl. In *ACM SIGGRAPH 2010 Posters*, SIGGRAPH '10, pages 135:1–135:1, New York, NY, USA, 2010. ACM. 11

[Dir13] J. Dirksen. *Learning Three.js: The JavaScript 3D Library for WebGL*. Packt Publishing Birmingham, 2013. 13

[Dro11] B. Drozdz. J3D - Unity3D to Three.js exporter. URL `https://github.com/drojdjou/J3D`, 2011. 11

[EAB14] A. Evans, J. Agenjo, and J. Blat. Web-based visualisation of on-set point cloud data. In *Proceedings of the 11th European Conference on Visual Media Production*, CVMP '14, pages 10:1–10:8, New York, NY, USA, 2014. ACM. 15

[Edi16] Edition Topoi. Collection Repository. URL `http://repository.edition-topoi.org`, 2016. 119, 120, 121, 128, 130, 131, 136

[EEKI+11] W. Eastcott, D. Evans, V. Kalpias-Illias, J. Rooney, and M. Mihejevs. PlayCanvas. URL `https://playcanvas.com`, 2011. 24, 32, 39, 44, 51

[Epi14] Epic Games. Unreal Engine. URL `https://www.unrealengine.com`, 2014. 43, 48, 51

[ERB+14] A. Evans, M. Romeo, A. Bahrehmand, J. Agenjo, and J. Blat. 3d graphics on the web: A survey. *Computers & Graphics*, 41(0):43 – 61, 2014. 12

[Exo13] Exocortex Technologies. Clara.io. URL `https://clara.io`, 2013. 17, 33, 42

[Fra09a] Fraunhofer. Instant Reality. URL `http://www.instantreality.org`, 2009. 13, 22

[Fra09b] Fraunhofer. X3DOM. URL `http://https://www.x3dom.org`, 2009. 13, 57, 66, 67

[FS93] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 247–254, New York, NY, USA, 1993. ACM. 63

[GAW09] I. J. Grimstead, N. J. Avis, and D. W. Walker. Rave: the resource-aware visualization environment. *Concurrency and Computation: Practice and Experience*, 21(4):415–448, 2009. 9

[GB11] G. Graßhoff and C. Berndt. *Die Entasis der Säulen des Pantheon.* Berlin: Exzellenzcluster 264 Topoi, 2011. 121

[GCPDB14] F. Ganovelli, M. Corsini, S. Pattanaik, and M. Di Benedetto. *Introduction to Computer Graphics: a Practical Learning Approach.* Computer Graphics, Geometric Modeling, and Animation Series. CRC Press, Chapman & Hall/CRC, first edition, October 2014. 31

[Geo12] Geoweb3d Inc. Geoweb3d. URL `http://www.geoweb3d.com`, 2012. 49

[Geo14] GeorgiaTech. The PLY File Format. URL `http://www.cc.gatech.edu/projects/large_models/ply.html`, 2014. 62

[GMR+12] E. Gobbetti, F. Marton, M. B. Rodriguez, F. Ganovelli, and M. Di Benedetto. Adaptive quad patches: An adaptive regular structure for web distribution and adaptive rendering of 3d models. In *Proceedings of the 17th International Conference on 3D Web Technology*, Web3D '12, pages 9–16, New York, NY, USA, 2012. ACM. 14

[Goo09] Google. O3D. URL `https://code.google.com/archive/p/o3d`, 2009. 10

[Goo11a] Google. Google Earth. URL `https://www.google.com/earth`, 2011. 32, 49

[Goo11b] Google. Google Maps. URL `https://enterprise.google.com/maps`, 2011. 49, 112

[Goo11c] Google. WebGL Loader. URL `https://code.google.com/p/webgl-loader`, 2011. 22, 65, 66, 67

[Goo12] Goo Technologies. Goo Create. URL `https://github.com/GooTechnologies/goojs`, 2012. 41, 51

[Gra14] Gravity Sketch Limited. Gravity Sketch. URL `https://www.gravitysketch.com`, 2014. 36

[Gra16] GraphiTech. Geo Browser 3D. URL `http://geobrowser3d.com`, 2016. 49

[GS12] F. Ganovelli and R. Scopigno. Ocme: out-of-core mesh editing made practical. *IEEE Computer Graphics and Applications*, 32(3):46–58, May/June 2012. 63

[GVB+15] J. Gaillard, A. Vienne, R. Baume, F. Pedrinis, A. Peytavie, and G. Gesquière. Urban data visualisation in a web browser. In *Proceedings of the 20th International Conference on 3D Web Technology*, Web3D '15, pages 81–88, New York, NY, USA, 2015. ACM. 22

[Han97] C. Hand. A survey of 3d interaction techniques. *Comput. Graph. Forum*, 16:269–281, 1997. 30, 35

[HB15] M. Hildebrandt and J. Bohnet. Seerene. URL `https://www.seerene.com`, 2015. 49

[Her91] Heritage Council. Discovery Programme. URL `http://www.discoveryprogramme.ie`, 1991. 118

[Hic04] I. Hickson. Extending HTML. 2004. 8

[HKBR+14] D. Haehn, S. Knowles-Barley, M. Roberts, J. Beyer, N. Kasthuri, J. Lichtman, and H. Pfister. Design and evaluation of interactive proofreading tools for connectomics. *IEEE Transactions on Visualization and Computer Graphics*, 20:2466–2475, 2014. 22

[HLL+13] B. Houston, W. Larsen, B. Larsen, J. Caron, N. Nikfetrat, C. Leung, J. Silver, H. Kamal-Al-Deen, P. Callaghan, R. Chen, and T. McKenna. Clara.io: full-featured 3d content creation for the web and cloud era. In *SIGGRAPH Studio Talks*, 2013. 17

[Hop96] H. Hoppe. Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 99–108, New York, NY, USA, 1996. ACM. 14, 63

[Hou15] B. Houston. ThreeKit. URL `https://threekit.com`, 2015. 23, 50

[HRA+14] D. Haehn, N. Rannou, B. Ahtam, E. Grant, and R. Pienaar. Neuroimaging in the browser using the x toolkit. In *Front. Neuroinform. Conference Abstract: 5th INCF Congress of Neuroinformatics*, 2014. 21

[HTC16] HTC. Vive. URL `https://www.vive.com`, 2016. 37

[HUM05] HUMUSOFT. Orbisnap. URL `http://www.orbisnap.com`, 2005. 13

[IN13] D. Iborra and V. Nordstrom. Cl3ver. URL `https://www.cl3ver.com`, 2013. 39, 40

[JAG11] S. Jourdain, U. Ayachit, and B. Geveci. Paraviewweb: A web framework for 3d visualization and data processing. 2011. 9

[Jan11] J. Jankowski. A taskonomy of 3d web use. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 93–100, New York, NY, USA, 2011. ACM. 33

[JBG11] Y. Jung, J. Behr, and H. Graf. X3dom as carrier of the virtual heritage. 2011. 12

[JD12a] J. Jankowski and S. Decker. A dual-mode user interface for accessing 3d content on the world wide web. 2012. 33, 39

[JD12b] J. Jankowski and S. Decker. On the design of a dual-mode user interface for accessing 3d content on the world wide web. In *Int. J. Hum.-Comput. Stud.*, 2012. 33

[JFM+08] S. Jourdain, J. Forest, C. Mouton, B. Nouailhas, G. Moniot, F. Kolb, S. Chabridon, M. Simatic, Z. Abid, and L. Mallet. Sharex3d, a scientific collaborative 3d viewer over http. In *Proceedings of the 13th International Symposium on 3D Web Technology*, Web3D '08, pages 35–41, New York, NY, USA, 2008. ACM. 12

[JH13] J. Jankowski and M. Hachet. A Survey of Interaction Techniques for Interactive 3D Environments. In M. Sbert and L. Szirmay-Kalos, editors, *Eurographics 2013 - State of the Art Reports*. The Eurographics Association, 2013. 30

[JH15] J. Jankowski and M. Hachet. Advances in interaction with 3d environments. *Computer Graphics Forum*, 34(1):152–190, 2015. 30

[Jmo13] Jmol Development Team. JSmol - JavaScript-Based Molecular Viewer From Jmol. URL `http://sourceforge.net/projects/jsmol`, 2013. 35, 49

[Jog04] JogAmp. JOGL - Java OpenGL. URL `http://jogamp.org/jogl/www`, 2004. 9

[Joh07] T. Johansson. Taking the Canvas to Another Dimension. URL `https://web.archive.org/web/20071117170113/http://my.opera.com/timjoh/blog/2007/11/13/taking-the-canvas-to-another-dimension`, 2007. 10

[JRS+13] J. Jankowski, S. Ressler, K. Sons, Y. Jung, J. Behr, and P. Slusallek. Declarative integration of interactive 3d graphics into the world-wide web: Principles, current approaches, and research agenda. In *Proceedings of the 18th International Conference on 3D Web Technology*, Web3D '13, pages 39–45, New York, NY, USA, 2013. ACM. 12

[JvL16] B. Jenny, B. Šavrič, and J. Liem. Real-time raster projection for web maps. *International Journal of Digital Earth*, 9(3):215–229, 2016. 22

[Kam04] M. Kamburelis. View3dscene. URL `https://castle-engine.sourceforge.io/view3dscene.php`, 2004. 13

[Kay10] L. Kay. SceneJS. URL `http://scenejs.org`, 2010. 26

[KH13] M. Kazhdan and H. Hoppe. Screened Poisson Surface Reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13, July 2013. 99

[Khr03] Khronos Group. OpenGL ES - The Standard for Embedded Accelerated 3D Graphics. URL `https://www.khronos.org/opengles`, 2003. 10

[Khr09] Khronos Group. WebGL - OpenGL ES for the Web. URL `https://www.khronos.org/webgl`, 2009. 8, 55, 77

[Khr15a] Khronos Group. glTF - GL Transmission Format. URL `https://www.khronos.org/gltf`, 2015. 15

[Khr15b] Khronos Group. WebGL section at SIGGRAPH 2015. URL `https://www.khronos.org/news/events/2015-siggraph`, 2015. 29

[KMF+08] A. Khan, I. Mordatch, G. Fitzmaurice, J. Matejka, and G. Kurtenbach. Viewcube: A 3d orientation indicator and controller. In *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, I3D '08, pages 17–25, New York, NY, USA, 2008. ACM. 34

[KRS+13] F. Klein, D. Rubinstein, K. Sons, F. Einabadi, S. Herhut, and P. Slusallek. Declarative ar and image processing on the web with xflow. In *Proceedings of the 18th International Conference on 3D Web Technology*, Web3D '13, pages 157–165, New York, NY, USA, 2013. ACM. 13

[KSR+12] F. Klein, K. Sons, D. Rubinstein, S. Byelozyorov, S. John, and P. Slusallek. Xflow - declarative data processing for the web. In *Proceedings of the 17th International Conference on Web 3D Technology*, Los Angeles, California, 2012. 13

[KSRS13] F. Klein, K. Sons, D. Rubinstein, and P. Slusallek. Xml3d and xflow: Combining declarative 3d for the web with generic data flows. *IEEE Computer Graphics and Applications*, 33(5):38–47, Sept 2013. 13

[KTL+04] D. Koller, M. Turitzin, M. Levoy, M. Tarini, G. Croccia, P. Cignoni, and R. Scopigno. Protected interactive 3d graphics via remote rendering. *ACM Trans. on Graphics*, 23(3):695–703, Aug 2004. 63

[Kub13] Kubity. Kubity. URL `https://www.kubity.com`, 2013. 42

[Kwa15] E. Kwan. Touch with WebGL and Leap Motion. URL `https://developer-archive.leapmotion.com/gallery/touch-with-webgl-leap-motion`, 2015. 36

[LCD13] G. Lavoué, L. Chevalier, and F. Dupont. Streaming compressed 3d data on the web using javascript and webgl. In *Proceedings of the 18th International Conference on 3D Web Technology*, Web3D '13, pages 19–27, New York, NY, USA, 2013. ACM. 14, 22, 65

[LCD14] G. Lavoué, L. Chevalier, and F. Dupont. Progressive streaming of compressed 3d graphics in a web browser. In *ACM SIGGRAPH 2014 Talks*, SIGGRAPH '14, pages 43:1–43:1, New York, NY, USA, 2014. ACM. 14, 22

[LCD+15] C. Leoni, M. Callieri, M. Dellepiane, D. P. O'donnell, R. Rosselli Del Turco, and R. Scopigno. The dream and the cross: A 3d scanning project to bring 3d content in a digital edition. *J. Comput. Cult. Herit.*, 8(1):5:1–5:21, February 2015. 31

[LD13] C. Lehmann and J. Döllner. Annotating 3d content in interactive, virtual worlds. In *Web3D*, 2013. 41

[Leo15] Leopoly LTD. Leopoly. URL `https://leopoly.com`, 2015. 51

[Leu08] C. Leung. C3DL - Canvas 3D JS Library. URL `https://github.com/cathyatseneca/c3dl`, 2008. 10

[LJBA13] M. Limper, Y. Jung, J. Behr, and M. Alexa. The pop buffer: Rapid progressive clustering by geometry quantization. *Computer Graphics Forum*, 32(7):197–206, 2013. 15, 57, 65

[LLD12] H. Lee, G. Lavoué, and F. Dupont. Rate-distortion optimization for progressive compression of 3d mesh with color attributes. *Vis. Comput.*, 28(2):137–153, February 2012. 14, 63

[LO14] R. Lebrun and M. J. Orliac. MorphoMuseum. URL `http://morphomuseum.com`, 2014. 79

[LO16] R. Lebrun and M. J. Orliac. Morphomuseum: An online platform for publication and storage of virtual specimens. *The Paleontological Society Papers*, 22:183195, 2016. 79

[LTBF14] M. Limper, M. Thöner, J. Behr, and D. W. Fellner. Src - a streamable format for generalized web-based 3d data transmission. In *Proceedings of the 19th International ACM Conference on 3D Web Technologies*, Web3D '14, pages 35–43, New York, NY, USA, 2014. ACM. 15, 22, 57

[Lue03] D.P. Luebke. *Level of Detail for 3D Graphics*. The Morgan Kaufmann Series in Computer Graphics Series. Morgan Kaufmann Publishers, 2003. 14

[LWJ07] LWJGL. Lightweight Java Game Library. URL `https://www.lwjgl.org`, 2007. 9

[LWS+13] M. Limper, S. Wagner, C. Stein, Y. Jung, and A. Stork. Fast delivery of 3d web content: A case study. In *Proceedings of the 18th International Conference on 3D Web Technology*, Web3D '13, pages 11–17, New York, NY, USA, 2013. ACM. 14

[Mal12] D. Malyshau. Kri-Web - Functional 3D Engine for the Web. URL `https://code.google.com/archive/p/kri-web`, 2012. 11

[Mal13] L. Malomo. Generalized trackball and 3d touch interaction. *Master thesis, Università degli Studi di Pisa*, 2013. 36, 124

[MAT02] MATLAB. Simulink 3D Animation. URL https://www.mathworks.com/products/3d-animation.html, 2002. 13

[MGW01] T. Malzbender, D. Gelb, and H.J. Wolters. Polynomial texture maps. In *SIGGRAPH*, 2001. 113

[Mic96] Microsoft Corporation. ActiveX. URL https://msdn.microsoft.com/en-us/library/aa751972(VS.85).aspx, 1996. 9

[Mic07] Microsoft Corporation. Silverlight. URL https://www.microsoft.com/silverlight, 2007. 9

[MJM13] R. K. Mueller, Gay. J., and M. Moissette. OpenJsCAD. URL https://openjscad.org, 2013. 50

[MKB+15] F. Mwalongo, M. Krone, M. Becher, G. Reina, and T. Ertl. Remote visualization of dynamic molecular data using webgl. In *Proceedings of the 20th International Conference on 3D Web Technology*, Web3D '15, pages 115–122, New York, NY, USA, 2015. ACM. 22

[MKRE16] F. Mwalongo, M. Krone, G. Reina, and T. Ertl. State-of-the-Art Report in Web-based Visualization. *Computer Graphics Forum*, 2016. 14

[Moz07] Mozilla. Canvas 3D. URL https://wiki.mozilla.org/Canvas:3D, 2007. 10

[Moz15] Mozilla. A-Frame. URL https://aframe.io, 2015. 13, 39, 44

[Moz16a] Mozilla. Device Motion Event. URL https://developer.mozilla.org/en-US/docs/Web/API/DeviceMotionEvent, 2016. 37

[Moz16b] Mozilla. Device Orientation Events. URL https://developer.mozilla.org/en-US/docs/Web/API/DeviceOrientationEvent, 2016. 37

[Moz16c] Mozilla. WebVR. URL https://developer.mozilla.org/en-US/docs/Web/API/WebVR_API, 2016. 37

[MSY13] L. Matheson, N. Schwinghamer, and A. Yanes. Pinshape. URL https://pinshape.com, 2013. 17, 50

[Mus15] Musum National d'Histoire Naturelle - Paris. Cabinet De Curiosites 3D. URL http://cabinetdecuriosites3d.mnhn.fr/en, 2015. 79

[MyM13] MyMiniFactory. MyMiniFactory. URL https://www.myminifactory.com, 2013. 25, 50

[NAS15] NASA - Jet Propulsion Laboratory. Experience Curiosity. URL `https://eyes.nasa.gov/curiosity`, 2015. 29

[NJ11] M. Nobel-Jørgensen. KickJS - A WebGL game engine for modern web-browsers. URL `http://www.kickjs.org`, 2011. 11, 51

[NO87] G. M. Nielson and Dan R. Olsen, Jr. Direct manipulation techniques for 3d objects using 2d locator devices. In *Proceedings of the 1986 Workshop on Interactive 3D Graphics*, I3D '86, pages 175–182, New York, NY, USA, 1987. ACM. 32

[Oct06] Octaga Visual Solutions. Octaga Player. URL `http://www.octagavs.com/solutions/web`, 2006. 13

[Ocu16] Oculus VR. Ocolus Rift. URL `https://www.oculus.com/rift`, 2016. 37

[Pac06] G. A. Pachikov. Cortona 3D. URL `http://www.cortona3d.com`, 2006. 13

[PCC+10] G. Palma, M. Corsini, P. Cignoni, R. Scopigno, and M. Mudge. Dynamic shading enhancement for reflectance transformation imaging. *JOCCH*, 3:6:1–6:20, 2010. 113

[PCD+15a] M. Potenziani, M. Callieri, M. Dellepiane, M. Corsini, F. Ponchio, and R. Scopigno. 3dhop: 3d heritage online presenter. *Int. Journ. Computers & Graphics*, 52:129–141, Nov 2015. 55, 133, 139

[PCD+15b] M. Potenziani, M. Callieri, M. Dellepiane, M. Corsini, F. Ponchio, and R. Scopigno. 3dhop: una piattaforma flessibile per la pubblicazione e visualizzazione su web dei risultati di digitalizzazioni 3d. *Archeomatica*, 6(4), Dec 2015. 139

[PCDS18] M. Potenziani, M. Callieri, M. Dellepiane, and R. Scopigno. Publishing and consuming 3d content on the web, a survey. *Foundations and Trends in Computer Graphics and Vision*, pages 89 (submitted paper, under review), 2018. 140

[PD15] F. Ponchio and M. Dellepiane. Fast decompression for web-based view-dependent 3d rendering. In *Web3D 2015. Proceedings of the 20th International Conference on 3D Web Technology*, pages 199–207. ACM, 2015. 15, 62, 64

[PD16] F. Ponchio and M. Dellepiane. Multiresolution and fast decompression for optimal web-based rendering. *Graphical Models*, 88:1 – 11, 2016. 15, 62, 65, 68

[Per09] M. Persson. Minecraft. URL `https://minecraft.net`, 2009. 9

[PFDS16] M. Potenziani, B. Fritsch, M. Dellepiane, and R. Scopigno. Automating Large 3D Dataset Publication in a Web-Based Multimedia Repository. In Giovanni Pintore and Filippo Stanco, editors, *Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference*. The Eurographics Association, 2016. 119, 136, 139

[Pin11] C. Pinson. OSGJS. URL `http://osgjs.org`, 2011. 26

[PPD+15] F. Ponchio, M. Potenziani, M. Dellepiane, M. Callieri, and R. Scopigno. The ariadne visual media service. In *CAA 2015*, page 12, 2015. 108, 136, 139

[PS97] E. Puppo and R. Scopigno. Simplification, lod and multiresolution principles and applications. 1997. 14

[PSP+12] G. Palma, E. Siotto, M. Proesmans, M. Baldassarri, C. Baracchini, S. Batino, and R. Scopigno. Telling the story of ancient coins by means of interactive rti images visualization. In *CAA 2012 Conference Proceeding*, pages ?? – ?? Pallas Publications - Amsterdam University Press (AUP), 2012. 116

[PWWS03] M. Patel, M. White, K. Walczak, and P. Sayd. Digitisation to presentation - building virtual museum exhibitions. In *VVG*, 2003. 12

[Rag94] D. Raggett. Extending WWW to support platform independent virtual reality. *Technical Report*, 1994. 8, 12, 32

[RB93] J. Rossignac and P. Borrel. Multi-resolution 3d approximations for rendering complex scenes. In Bianca Falcidieno and TosiyasuL. Kunii, editors, *Modeling in Computer Graphics*, IFIP Series on Computer Graphics, pages 455–465. Springer Berlin Heidelberg, 1993. 63

[Res06] J. Resig. jQuery - Write less, do more. URL `https://jquery.com`, 2006. 77

[RH15] A. S. Rose and P. W. Hildebrand. Ngl viewer: a web application for molecular visualization. *Nucleic Acids Research*, 43(W1):W576–W579, 2015. 22

[Ric02] J. F. Richardsoon. SimVRML. URL `https://sourceforge.net/projects/simvrml`, 2002. 13

[RMBB+13] B. C. Russell, R. Martin-Brualla, D. J. Butler, S. M. Seitz, and L. S. Zettlemoyer. 3d wikipedia: using online text to automatically label and navigate reconstructed geometry. *ACM Trans. Graph.*, 32:193:1–193:10, 2013. 40

[RWW14] B. Resch, R. Wohlfahrt, and C. Wosniok. Web-based 4d visualization of marine geo-data using webgl. *Cartography and Geographic Information Science*, 41(3):235–247, 2014. 22

[SC92] P. S. Strauss and R. Carey. An object-oriented 3d graphics toolkit. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 341–349, New York, NY, USA, 1992. ACM. 32

[SCD+17] R. Scopigno, M. Callieri, M. Dellepiane, F. Ponchio, and M. Potenziani. Delivering and using 3d models on the web: are we ready? *Virtual Archaeology Review*, 8(17):1–9, 2017. 139

[Sch13] M. Schuetz. Potree. URL `http://potree.org`, 2013. 27, 34, 48

[Sch15] M. Schuetz. Rendering large point clouds in web browsers. In *Central European Seminar on Computer Graphics 2015*, 2015. 15

[SD13] R. Scopigno and M. Dellepiane. Ariadne infrastructure for Multimedia data: Matching technologies and user needs. In *ARIADNE Workshop Report*, 2013. 109

[SDC+15] E. Siotto, M. Dellepiane, M. Callieri, R. Scopigno, C. Gratziu, A. Moscato, L. Burgio, S. Legnaioli, G. Lorenzetti, and V. Palleschi. A multidisciplinary approach for the study and the virtual reconstruction of the ancient polychromy of Roman sarcophagi. *Journal of Cultural Heritage*, (16):307–314, 2015. 98

[SG05] E. Shaffer and M. Garland. A multiresolution representation for massive meshes. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):139–148, March 2005. 63

[SH94] B. R. Schatz and J. B. Hardin. Ncsa mosaic and the world wide web: Global hypermedia protocols for the internet. *Science*, 265(5174):895–901, 1994. 8

[SH15] S. Shi and C. Hsu. A survey of interactive remote rendering systems. *ACM Comput. Surv.*, 47:57:1–57:29, 2015. 14

[Sha13] Shapeways. Shapeways. URL `https://www.shapeways.com`, 2013. 25, 50

[Shn83] B. Shneiderman. Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57–69, Aug 1983. 33

[Sho92] K. Shoemake. Arcball: A user interface for specifying three-dimensional orientation using a mouse. In *Proceedings of the Conference on Graphics Interface '92*, pages 151–156, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc. 31

[SIS02] P. S. Strauss, P. Issacs, and J. Shrag. The design and implementation of direct manipulation in 3d: Siggraph 2002 course notes. 2002. 32

[Ske14] Sketchfab. Sketchfab. URL `https://sketchfab.com`, 2014. 24, 28, 29, 40, 44, 48, 57, 65, 110

[SKR+10] K. Sons, F. Klein, D. Rubinstein, S. Byelozyorov, and P. Slusallek. Xml3d: Interactive 3d graphics for the web. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 175–184, New York, NY, USA, 2010. ACM. 13

[Smi11] Institution Smithsonian. Smithsonian X3D. URL `http://3d.si.edu`, 2011. 28, 34, 35, 48

[Son10] Sons, K. and Slusallek, P. XML3D. URL `http://xml3d.org`, 2010. 13, 58

[SP11] Z. Smith and B. Pettis. Thingiverse. URL `https://www.thingiverse.com`, 2011. 25, 44, 50

[SPPS15] E. Siotto, G. Palma, M. Potenziani, and R. Scopigno. Digital study and web-based documentation of the colour and gilding on ancient marble artworks. *2015 Digital Heritage*, 1:239–246, 2015. 97, 136, 139

[SPPS18] E. Siotto, G. Palma, M. Potenziani, and R. Scopigno. A web-based system for data integration and visualization of the ancient colour. In P. Liverani, editor, *Proceedings of 7th Round Table on Polychromy in Ancient Sculpture and Architecture*, Florence 2015, pages 183–188 (in press), 2018. 139

[SSK+13] K. Sons, C. Schlinkmann, F. Klein, D. Rubinstein, and P. Slusallek. xml3d.js: Architecture of a polyfill implementation of xml3d. In *2013 6th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*, pages 17–24, March 2013. 13

[SSS14] J. Sutter, K. Sons, and P. Slusallek. Blast: A binary large structured transmission format for the web. In *Proceedings of the 19th International ACM Conference on 3D Web Technologies*, Web3D '14, pages 45–52, New York, NY, USA, 2014. ACM. 15

[Sta15] Stackgl. Stackgl. URL `http://stack.gl`, 2015. 26

[Ste98] J. A. Stewart. FreeWRL. URL `http://freewrl.sourceforge.net`, 1998. 13

[Sun98] Sun Microsystems. JAVA3D - The Java 3D API. URL `http://www.oracle.com/technetwork/articles/javase/index-jsp-138252.html`, 1998. 9

[SYCK16] D. Seo, B. Yoo, J. Choi, and H. Ko. Webizing 3d contents for super multiview autostereoscopic displays with varying display profiles. In *Proceedings of the 21st International Conference on Web3D Technology*, Web3D '16, pages 155–163, New York, NY, USA, 2016. ACM. 36

[SYK15] D. Seo, B. Yoo, and H. Ko. Webized 3d experience by html5 annotation in 3d web. In *Web3D*, 2015. 41

[Thr13] Threeding. Threeding. URL `https://www.threeding.com`, 2013. 25, 50

[Top05] Topoi. Excellence Cluster. URL `http://www.topoi.org`, 2005. 120

[Tos12] M. Toschlog. Parallax. URL `http://parallax3d.org`, 2012. 37

[Tri14] Triumph LLC. Blend4web. URL `https://www.blend4web.com`, 2014. 29, 30, 39, 44, 51

[Tur09] Turbulenz. Turbulenz. URL `http://biz.turbulenz.com`, 2009. 51

[Ube15] Uber. Deck.gl. URL `https://uber.github.io/deck.gl`, 2015. 49

[Uni96] University of York. Archaeology Data Service. URL `http://archaeologydataservice.ac.uk`, 1996. 118

[Uni05] Unity Technologies. Unity3D. URL `https://unity3d.com`, 2005. 11, 26, 32, 43, 47, 48, 51, 56, 65

[Uni09] University College London (UCL). 3D Petrie Museum. URL `http://www.ucl.ac.uk/3dpetriemuseum`, 2009. 31, 48

[Uni11] University of Applied Sciences Northwestern Switzerland (FHNW). OpenWebGlobe. URL `https://github.com/OpenWebGlobe`, 2011. 49

[Uni14] University of Cape Town - Division of Geomatics. The Zamani Project. URL `http://zamaniproject.org`, 2014. 79

[Uni16] Universidad de la Repblica - Paleobiology Lab (FCIEN). Megafauna 3D. URL http://www.megafauna3d.org/en, 2016. 79

[vD97] A. van Dam. Post-wimp user interfaces. *Commun. ACM*, 40(2):63–67, February 1997. 35

[Vir15] Virtual Heritage Lab. Aton Front-End. URL http://osiris.itabc.cnr.it/scenebaker/index.php/projects/aton, 2015. 35

[Vis05] Visual Computing Lab. MeshLab - An open source, portable, and extensible system for the processing and editing of unstructured 3D triangular meshes. URL http://www.meshlab.net, 2005. 88, 99, 125

[Vis10] Visual Computing Lab. SpiderGL - 3D Graphics for Next-Generation WWW. URL http://vcg.isti.cnr.it/spidergl, 2010. 11, 55, 63, 77

[Vis13] Visual Computing Lab. Nexus - Multiresolution Visualization. URL http://vcg.isti.cnr.it/nexus, 2013. 63

[Vis14a] Visual Computing Lab. 3DHOP - 3D Heritage Online Presenter. URL https://3dhop.net, 2014. 55, 56, 60, 66, 67, 70, 73, 75, 77, 80, 134, 140

[Vis14b] Visual Computing Lab. MeshLabJS. URL http://www.meshlabjs.net, 2014. 42

[Vis15a] Visionary Cross. The Visionary Cross Project. URL http://vcg.isti.cnr.it/cross, 2015. 31

[Vis15b] Visual Computing Lab. ARIADNE - Visual Media Service. URL http://visual.ariadne-infrastructure.eu, 2015. 108, 109, 111, 112, 114, 116, 117, 118, 136

[Vis15c] Visual Computing Lab. Jackson Pollock: Alchemy in 3D. URL http://vcg.isti.cnr.it/alchemy, 2015. 84, 93, 95, 96, 135, 140

[Vis15d] Visual Computing Lab. The Annona Sarcophagus. URL http://vcg.isti.cnr.it/roman-sarcophagi/annona-sarcophagus, 2015. 97, 100, 102, 103, 104, 105, 135

[Viz14] Vizor. Patches. URL https://patches.vizor.io, 2014. 37

[Vox13] Voxel.js. Voxel.js. URL http://voxeljs.com, 2013. 51

[Wal11] E. Wallace. LightGL - A lightweight WebGL library. URL https://github.com/evanw/lightgl.js, 2011. 11, 17, 26

[WCW06] K. Walczak, W. Cellary, and M. White. Virtual museum exhibitions. *IEEE Computer*, 39:93–95, 2006. 12

[Web04] Web3D Consortium. What is X3D Graphics. URL `http://www.web3d.org/what-x3d-graphics`, 2004. 8, 12, 57

[Wil11] J. Wilhelmy. Inka3D. URL `http://www.inka3d.com`, 2011. 11

[Wit12] S. Wittens. MathBox. URL `https://gitgud.io/unconed/mathbox`, 2012. 41

[WO90] C. Ware and S. Osborne. Exploration and virtual camera control in virtual three dimensional environments. In *Proceedings of the 1990 Symposium on Interactive 3D Graphics*, I3D '90, pages 175–183, New York, NY, USA, 1990. ACM. 31

[WS06] M. Wimmer and C. Scheiblauer. Instant points: Fast rendering of unprocessed point clouds. In *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, SPBG'06, pages 129–137, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association. 63

[WWV+10] C. A. Wingrave, B. Williamson, P. Varcholik, J. Rose, A. Miller, E. Charbonneau, J. N. Bott, and J. J. LaViola. The wiimote and beyond: Spatially convenient devices for 3d user interfaces. *IEEE Computer Graphics and Applications*, 30:71–85, 2010. 36

[WWWC04] R. Wojciechowski, K. Walczak, M. White, and W. Cellary. Building virtual and augmented reality museum exhibitions. In *Web3D*, 2004. 12

[XTK12] XTK Developers. X Toolkit API. URL `https://github.com/xtk`, 2012. 21, 49

[ZCGC11] F. Zollo, L. Caprini, O. Gervasi, and A. Costantini. X3dmms: An x3dom tool for molecular and material sciences. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 129–136, New York, NY, USA, 2011. ACM. 12, 49

[ZFF15] P. Zuspan, J. Finkelstein, and M. Finkelstein. Kokowa. URL `https://www.kokowa.co`, 2015. 42

[ZHXJ10] Y. Zhang, X. Hua, J. Xiaohong, and S. Jiaoying. A Survey of Simplification and Multiresolution Techniques for Massive Meshes: A Survey of Simplification and Multiresolution Techniques for Massive Meshes. *Journal of Computer-aided Design & Computer Graphics*, 22:559–568, 2010. 63

[Zip10] A. Zipf. OSM-3D. URL http://www.osm-3d.org, 2010. 9