DELIVERABLE REPORT

D 5.4

Document identifier:	**V-Must.net - D 5.4**
Due Date of Delivery to EC	Juanyary 2015
Actual Date of Delivery to EC	March 2015
Document date:	December 2014
Deliverable Title:	**VM evaluation on portability, tangible visualisation and interaction techniques: Methodological guidelines with respect to portability, usability and integration.**
Work package:	**WP 5**
Lead Beneficiary:	INRIA
Other Beneficiaries	FRAUNHOFER, CNR, CULTNAT
Authors:	Patrick Reuter, Xavier Granier [*Contributors*: Holger Graf, Massimiliano Corsini, Alfonsina Pagano, Mohamed Farouk]
Document status:	Version 1.1
Document link:	http://www.v-must.net/media_pubs/documents

Change Log

Version	**Date**	**Comment**	**Authors**
0.1	15/12/2015	**First version** Structure and Template	Patrick Reuter, Xavier Granier
0.2	20/01/2015	**Second version** Review of material and results	Patrick Reuter, Alfonsina Pagano, Holger Graf
0.3	10/02/2015	**Third version** Add Useability	Patrick Reuter, Alfonsina Pagano, Massimiliano Corsini
0.4	15/02/2015	**Fourth version** Add Showcases	Patrick Reuter, Alfonsina Pagano, Daniele Ferdani, Mohamed Farouk
0.5	28/02/2015	**Fifth version** Add VM assets, Portability Analysis	Patrick Reuter, Alfonsina Pagano, Daniele Ferdani, Holger Graf, Massimiliano Corsini, Mohamed Farouk
1.0	28/02/2015	**Final version** Final editing	Patrick Reuter, Holger Graf, Alfonsina Pagano

Table of Contents

Executive Summary

This deliverable represents the first iteration of an on-going set of activities aimed at identifying the best practices of virtual museums regarding portability, usability and integration.This defines a scaffold of criteria useful for the evaluation of portability (intended as multi-fruition modes), tangible visualisation and interaction techniques of the best examples carried out under the V-MusT.NET consortium.

The final goal is to produce guidelines and indications on how to create a good portable application or an effective tangible interactive application, disconnected from the physical location. Therefore, a cross-analysis on parameters such as "usability", "portability" and "integration" had been set up, taking advantages of recent experiences in the field of tangible and gesture-based interaction projects in immersive installations.

1. Usability and Portability under the V-Must.net framework

The aims of WP5, and in particular, the D5.4 tasks associated with the V-MusT.NET consist in investigating which are the criteria useful to design and implement an effective portable application taking care of its usability and integration inside the exposition context. Terms like "usability" and "integration" have been largely discussed in V-MusT.NET projects arriving sometime to a concrete definition and a reliable field of appliance.

This deliverable represents a preliminary attempt towards achieving these goals, and we envision to further develop a methodological guideline for a real evaluation of such criteria, making the Next Generation of Virtual Museums (NGVMs), core content of this European network, portable under certain conditions. In V-MusT.NET, we are considering different levels of portability:

- the first one sees an exhibition moved from one museum to an other with a minimal amount of issues and difficulties in terms of transportation and adaption;
- the second one implies the technical concepts and services used from one exhibition re-used in an other context.

The core focus of the work conducted in WP5 is directed towards understanding and influencing the integration of VM applications into the museum context, considering the roles of information architecture (content) and system development, in conceptual and practical ways.

Before start discussing of virtual museums setup, we need to introduce the theme. While the definition of the term "virtual museum" continues to evolve, the international council of museums (ICOM) adopted a single definition for "museum" at its 21st General Conference in Vienna, Austria, in 2007, which states:

> *A museum is a non-profit, permanent institution in the service of society and its development, open to the public, which acquires, conserves, researches, communicates and exhibits the tangible and intangible heritage of humanity and its environment for the purposes of education, study and enjoyment [1].*

This definition has been applied to museums for several years now, however, there is no indication of the role of technology in the museum. While technology may not yet be specifically mentioned in the current ICOM definition, it has the potential to play a significant role in the range of functions provided for and by museums. Within the context of V-MusT.NET, specifically in the work conducted in WP2 and in line with the most relevant actors in the digital heritage domain (see D2.1c), we elected in 2015 a final definition of the term "virtual museum":

> *[Vers. 1.5]*
> *A virtual museum is a digital entity that draws on the characteristics of a museum, in order to compliment, enhance, or augment the museum experience through personalization, interactivity and richness of content. Virtual museums can perform as the digital footprint of a physical museum, or can act independently, while maintaining the authoritative status as bestowed by ICOM in its definition of a museum. In tandem with the ICOM mission of a physical museum, the virtual museum is also committed to public access; to both the knowledge systems embedded in the collections and the systematic and coherent organisation of their display, and their long-term preservation.*

This definition is also available at: https://en.wikipedia.org/wiki/Virtual_museum.

1.1 Usability: V-MUXE scaffold

The concept of Usability has been largely discussed in WP2, specifically in D2.1, about terminology under V-MusT.NET and it has been deeply investigated within lots of projects and drafted in WP7. This term can be defined as the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use [ISO, 1998]. In our cases, VMs have been studied according to their interface elements, navigability, and content.
Likely, usability of an interface is associated to five parameters [ISO, 1998; Nielsen, 1993], derived directly from this definition:

- *easy to learn:* the user can get work done quickly with the system;
- *efficient to use*: once the user has learnt the system, a high level of productivity is possible;

- *easy to remember:* the casual user is able to return to using the system after some period without having to learn everything all over again;
- *few errors:* users do not make many errors during the use of the system or if they do so they can easily recover them;
- *pleasant to use*: users are subjectively satisfied by using the system; they like it.

In the V-MusT.NET consortium, the most of the projects saw an evaluation on design indicators starting from the usability of respective systems (and further on the setup of installations and their content). Indeed, usability studies includes methods for measuring usability, such as needs analysis or users interacting with such technologies/products and the study of the principles behind an object's perceived efficiency or aesthetical appareance. In human-computer interaction, usability studies the clarity with which the interaction with a digital product or a website (web usability) is designed.

When studying usability, relevant to take into account is the concept of “user experience”, especially when conducting addressed evaluations. Again in D2.1, user experience is defined by ISO [ISO 2009] which conceives it as "a person's perceptions and responses that result from the use or anticipated use of a product, system or service". In general, user experience (UX) is how a person feels when interfacing with a system. The system could be either a website interface or a web application, desktop installation or finally a mobile product. It is generally characterized by main features related to human-computer interaction (HCI) disciplines. Compared to many others, UX is newly-born: the term “user experience”, coined by Dr. Donald Norman, relies upon the concepts of usability, interaction design, visual studies and information architecture.

A first real attempt in fixing a scaffold of evaluative criteria out of usability research in V-MusT.NET project is the V-MUXE work done at the occasion of the international exhibition on Virtual Archaeology, Archeovirtual 2012. The V-MusT.NET provided a first list of design indicators required for the Next Generation of Virtual Museums (NGVMs). The methodology employed was based on a holistic approach using user observations, user profiling through interviews and evaluation studies with target users. For the occasion, we developed new questionnaires that account for the elevation of user experience (UX) variables deploying a synthesized set of (non-)instrumental qualities (e.g. utility, usability, motivation, emotions, etc.). The analysis of these qualities enabled us to compare different forms of installations and their impact on UX for different target groups as well as their impact on new design guidelines for NGVM.

What resulted is that features really affecting user experience like *utility*, *learnability, efficiency* and *stimulation* may increase or decrease according to the different levels of interactivity and immersivity. Generally, we can draft the following considerations:

- *Utility* is the majority of the time founded on (a) the consistency of the interfaces’ elements, (b) linearity and conciseness of contents and (c) a good mapping between users’ control and system feedback;
- *Learnability* sees connections with (a) good visibility of navigation indicators, (b) affordance of visual and textual information and (c) the consistent nature of contents delivered to users (transparency of information architecture);

- *Efficiency* takes advantages of (a) the rapid feedback given by the system to users and (b) the multiplicity of functions offered;
- *Stimulation* is influenced by (a) the integration of multi-media, which brings to pleasant aesthetical features of the system itself, (b) by the whole environment and (c) the easiness of interaction.

Specifically about mobile-based applications, which imply *gesture-based interaction*, mainly face issues on the visibility of system elements and navigability. This obviously reflects in the usability of those, and it might leave users disoriented by misleading information and confusing interfaces. In line with this, the same visual and textual indicators are influenced by consistency and users' orientation. Besides these aspects, VMs of this category are highly appreciated and considered user-friendly, given the widespread of such devices.

Emerging technologies like immersive headsets (e.g. Oculus rift), CAVE projects, thus between *device-based and natural interaction*, require practically a constant maintenance by the particular developers on site because of the necessity of a preliminary introduction to users (fitting of the headset, explanation on the interaction). Here, the feature of easiness of the system turns to be a subjective datum, as the user alone cannot know the potential of the interaction without the assistant. In general, these VMs are more suitable for visibility and aesthetical evaluations since interface and visual elements are the most interesting items highlighted by users (i.e. interest in multimedia functions, icons, customization of characters, just to name a few).

Concerning physically interactive applications, obviously relying upon a *natural interaction* augmented through immersive scenarios, they are appreciated for their immersiveness, the natural gestural environment recreated, the good feedback of the system - in sending back information about what actions need to be done and what are the tasks accomplished. Nevertheless, some efforts are suggested towards content visibility, clarity of interface's elements, and mapping between users' controls and system's effects. Again for this category, the presence of developers on site was essential.

Desktop-based applications generally present issues belonging to content and structural matters. The narrative thread, which lies beyond the content architecture of some V-Must.NET virtual museums, is usually appreciated by users. Some others more addressed to experts in the field of CH present meta-data and content modelling system. Obviously, the latter might have more problems of affordance of certain information, visibility , and aesthetical features. What would be worthy to further improve is the definition of the final addressees of each VM, because language (database mode), way of presenting data (formatting) and level of complexity of the system (multiplicity of functions) inevitably influence any evaluation of this category.

Touchscreen applications involve tactile interaction, and they are used in various types of VMs and differ in interaction concepts and content structures.

Last but not least, with tangible visualisation and interaction, the real physical world is augmented by coupling digital information to physical objects and environments. In Section 2, we present tangible visualisation and interaction in detail, and in Section 3, we provide a concrete case study.

For more details and further concrete analysis of virtual museums regarding user experience and usability, we refer to the V-MUXE publication [Gockel et al, 2013] that was established within the V-Must.NET consortium.

1.2 Portability: towards a definition for VMs

The V-MusT.NET consortium proposes the following definition for portability in D2.1 as referable to

> "*all the VM applications whose software or hardware components are not publicly available, whatever the reason (technical need or political choice) of such lack of availability, but which can be potentially fruited in any place at the same conditions, without a binding link between the application's physical placement and its content".*

The accentuation concerns the capability of a Virtual Museum of (a) being experienced in every situation/environment/setup, and (b) under the same contextual conditions. Furthermore, the autonomy of the content is also highlighted, given its freedom to choose the final location of the application's display, which may influence the overall information architecture and storytelling (when applicable). Portability can thus be seen as subcategory of the *sustainability* level, one of the key aspects the V-MusT.NET consortium is interested in, and that regards the future persistence and lifetime of applications and contents in cultural institutions.

Likely the concept of portability is strictly related to *flexibility*. More precisely, portability relates to the possibility of reusing or re-adapting computer hardware and/or software created for a specific VM in any other exhibitions and with different content. Note that portability should not be confounded with the *transmediality* concept ([see D7.4]: where content is independent from the hardware infrastructure - potentially, it can be rearranged within different media communication channels).

Consider **Admotum**, an example of a portable Virtual Museums. **Admotum** is a Natural Interaction Application that follows a game-like mechanism within the Keys2Rome exhibition (www.keys2rome.eu). It is used to reinforce the visitors' experience of a Keys2Rome visit: it enables the visitors to recognize objects coming from 4 different European museums, within their original contexts. This application has been deployed in all the four different venues of the Keys2Rome exhibition: Rome (Museum of the Trajan Market); Sarajevo (City Hall); Amsterdam (Allard Pierson Museum); Alexandria (Bibliotheca Alexandrina). The contents are the same, but they are personalized concerning the starting location, which is set to the museum the visitor happens to be (for example, if the visitor plays the game in Rome, he will start from roman scenarios). The adoption of a modular, customizable, multi-input system represents a flexible solution regarding the physical space that is available for the interaction: either natural interaction using the kinect sensor, or a desktop solution with a standard mouse-keyboard setup. The overall low-cost setup and the involved devices (Kinect, projector and a standard PC) are also adjustable for different room spaces and shapes, in addition of being easy to assemble and easy to re-use.

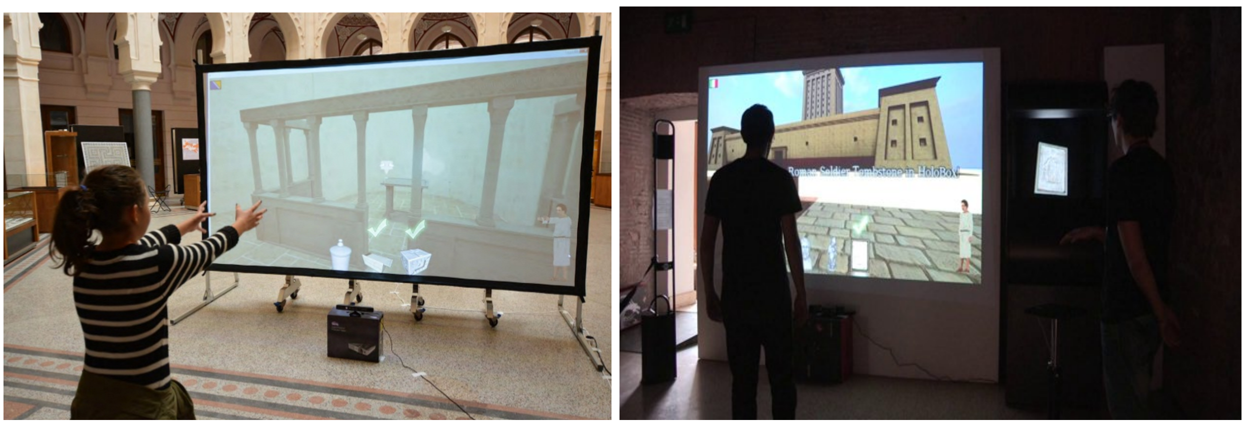

Fig.1 - Two different set ups of Admotum adapted in different Key2Rome venues.

Given the showcase presented above, we consider *portability* as a fundamental requirement that must be taken into account before designing a VM. Indeed, portability is required for supporting both temporary exhibitions, such as touring exhibitions, special events, expos that require infrastructures easily to carry and set up from time to time in different venues, and also permanent exhibitions when a proven technology fits to the museum/site contents. Below we list some consistent features of Portable VMs:

- *Simple setup*: easy and quick to assemble and calibrate
- *Adaptable design*: set up flexible and easy-fitting in the most cases
- *Portable equipment*: availability, weight, size and price of the hardware
- *Durable equipment*: solid technology and proven operating principles
- *Spare equipment*: easily to update and repair with easily obtainable spare parts

We will analyze a further example of a portable virtual museum in section 3, where we discuss the revealing flashlight as a case study of tangible visualisation and interaction.

2. Tangible visualization and new interaction paradigms

2.1 Introduction

When visitors enter a museum building, they come to live an experience that they would not have by their own means (as for example at home on the internet): museum visitors come to museums to see real physical cultural artifacts. Most often, the visitors are not allowed to touch cultural artifacts, not to mention to change their physical state. We advocate that VMs make it possible to enhance the perception of and interact with physical cultural artifacts. Ideally, since museum visitors often come in groups of people, they may discover cultural heritage together rather than living individual experiences.

In this section, we present two ways to improve the museum visitors' usability experience, namely tangible visualization and spatial interaction techniques.

2.2 Tangible visualization in the Cultural Heritage Domain

As according to Wu [Wu 2010], the visualization of virtual information “should escape the traditional constraint of the screen and embrace the physical environment to realize more tangible interactions for manipulating digital information.” Wu defines tangible visualization as *visualizations that exist in our physical environment.*

Augmented reality applications, where the user's view of the real-world is augmented by computer-generated information, are perfect candidates for tangible visualization.

As opposed to see-through augmented reality applications, where the display of a real scene on a screen (mobile phones, tablets, head-mounted displays, or glasses) is augmented by additional information, we rather advocate for using **spatial augmented reality** [Raskar et al. 1998]: the user's physical environment is augmented with images that are integrated directly in the real-world, and not simply on a display. This can be done by using a video-projector.

One of the major advantages is that since a display is not associated with each user, spatial augmented reality scales up naturally for numerous users, thus enabling community experiences and collocated collaboration.

Spatial augmented reality naturally is particularly interesting for cultural heritage applications: the virtual information layer integrates almost seamlessly, and non-3D experts may continue to reason in the real-world instead of an a computer screen.

2.3 Interaction techniques: tangible interaction and gesture-based interaction

In tangible visualization, as for example spatial augmented reality, the co-location of the visualization and interaction space enables efficient and natural interaction when using ***spatial user interaction***, such as tangible or gesture-based interaction. Spatial user interaction is a growing field as the sensing and display technologies are reaching a larger audience.
Input modalities in spatial user interaction can be gesture-based or use dedicated devices. A device, as involved in tangible user interfaces, makes it easier for the user to understand how to use the system since the device provides task-specific affordances. However, the involvement of user-manipulated fragile devices in museum exhibitions induces damage concerns.

In contrast, gesture-based input enables instantaneous interaction with the system. This kind of input generally relies on optical tracking using an RGB or depth camera. Another possibility would be the instrumentation of the environment with conductive surfaces, however, this is not well suited for cultural heritage artifacts, since again, direct touch manipulation is not adapted for fragile objects.

In addition to the choice of the input modality, one of the most important factors in spatial user interaction is the interaction technique itself, that is how the provided user input is actually mapped to the accomplishment of a task, and hence to the parameters of the system. In the following section, we present a tangible visualization and interaction technique that is suited to both input modalities: finger-gesture input or by using a prop device.

3. A case study: The Revealing Flashlight

3.1 Introduction

Cultural heritage artifacts often suffer from lost colorimetric or geometric detail due to aging effects such as erosion. The revealing flashlight [Ridel et al. 2014] was conceived within the V-must consortium; it is a new interaction and visualization technique in spatial augmented reality that helps to reveal the detail of cultural heritage artifacts. More precisely, the physical artifact is locally and interactively augmented by projecting virtual information directly on it. This virtual information can be anything, for example an interpretation of its original colours that might have been lost due to aging. It can also be an expressive, non-photo-realistic visualization that highlights geometric features that are hardly visible with the naked eye.

The revealing flashlight involves an interaction technique that simulates and improves the behavior of a flashlight: according to a specified position and orientation by finger gesture or a tangible six degrees-of-freedom device, the location to be augmented is determined. Moreover, additionally involved parameters can be specified.

The revealing flashlight can be used by museums to let visitors interactively discover ancient color information or geometric details of cultural artifacts. It can also be used by archaeologists, for example, to help decipher inscriptions in eroded stones.

3.2 Illustration and technical considerations

For an explicit illustration, consider Figure 3 that shows the revealing flashlight in action in the Keys2Rome exhibition in the Roma Imperial Fora Museum venue. There are four main items:

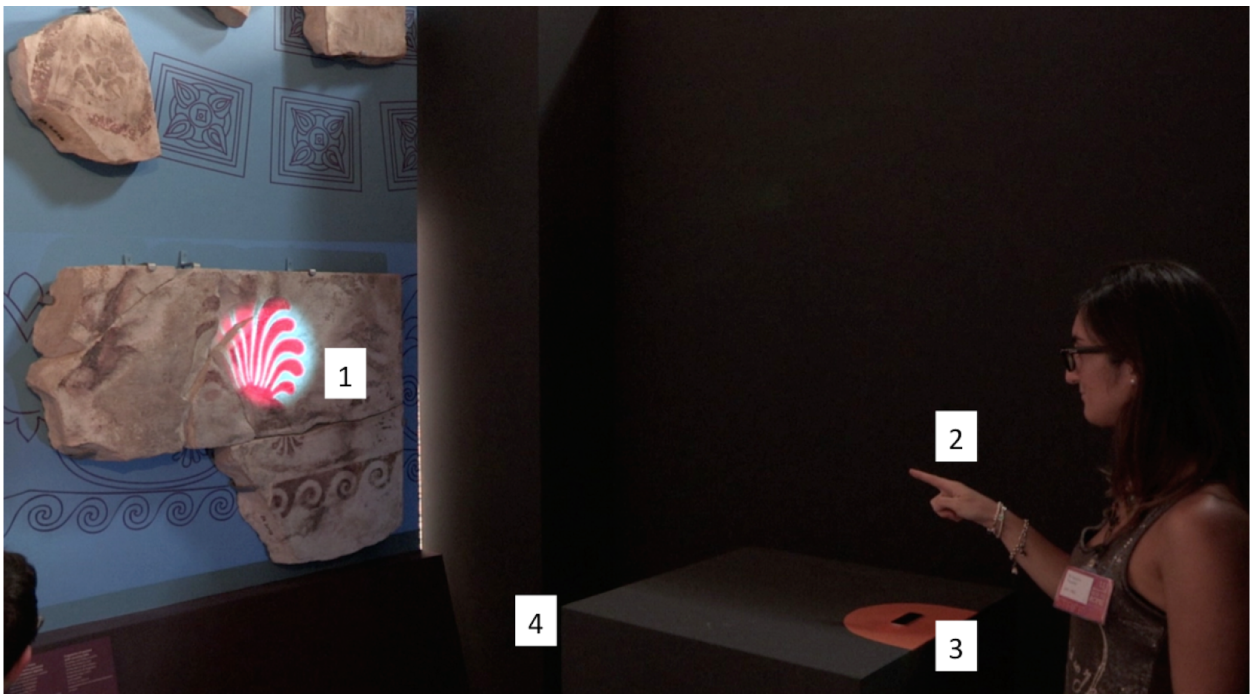

Fig. 2 - The revealing flashlight in the Roma Imperial Fora Museum.

For the inspection of a physical cultural artifact (item 1, the marble slabs), the user provides direct finger pointing input (item 2, the index): the finger's position and orientation are detected by an optical tracking device (item 3, a Leap Motion sensor). According to this position and orientation, ancient color information or geometric detail is superimposed directly at the pointed location on the artifact. This is done by a video-projector that is hidden in the desk (item 4). This installation brings back to life the ancient painted decoration interactively. Since the projector is invisible to the visitor, the interactive augmentation has a surprising and somehow magic effect. As the interaction metaphor reminds to the use of a traditional flashlight that, beyond illumination, reveals details of the cultural heritage artifact, the name of the interaction technique in spatial augmented reality is *the revealing flashlight*.

From a more technical point of view, the revealing flashlight requires a 3D model of the cultural artifact that can be acquired by any 3D acquisition technique such as laser range scanning or photo-grammetry. The color restoration or geometric detail analysis is actually done on the 3D model. In the museum setup, a previously calibrated video-projector has to be located with respect to the real artifact. During the interactive use, the viewing parameters of the 3D visualization are adjusted according to the finger gesture.

For a better understanding of the portability subsection below, note that the revealing flashlight consists of the following components:

- a cultural artifact
- a corresponding 3D model
- a tracking device
- a video-projector
- a computer
- a software
- and a parameterization of the software.

3.3 Evaluation of the revealing flashlight

The effectiveness, ease-of-use and ease-of-learning of the revealing flashlight was confirmed by exploratory user studies and concrete feedback of several public exhibitions.

For example, the revealing flashlight was on display in a 6 month-long exhibition in the Allard Pierson Museum in Amsterdam (the Netherlands) with a relief fragment from the wall of a Mastaba, a type of ancient Egyptian tomb. This fragment was originally colored, but all the original pigmentation has been lost. The revealing flashlight was used to interactively superimpose the lost colors (see Figure 4).

Fig. 3 - The revealing flashlight in the Allard Pierson Museum

In a 10 day user evaluation period, among the 42 subjects (21 men and 21 women, of all ages) that tested the revealing flashlight and replied to a questionnaire, 34 subjects (81%) found that it is a positive addition to the exhibition offer. It is also interesting to note how the subjects learned to use the revealing flashlight and how their attention was captured (Table 1).

I read the instructions.	*42%*
I tried until successful.	*23%*
I saw somebody else using it.	*15%*
A museum employee explained how to use it.	*15%*
I didn't manage to use it.	*6%*

Table 1 - How the visitors learned to use the revealing flashlight

3.4 Usability

Concerning the usability the revealing flashlight, recall the five usability parameters for interfaces according to [ISO, 1998; Nielsen, 1993] that we already named in Section 1: *easy to learn, efficient to use, easy to remember, few errors, pleasant to use*. In the following, we present our experience of our exploratory user studies with the revealing flashlight according to these five parameters :

- *easy to learn:* there are numerous ways to learn how to use the revealing flashlight (Table 1), and according to our experience, there is almost always a way that is adapted for a particular visitor that he can get work done quickly with the system. And beyond: we observed frequently that museum visitors that used the revealing flashlight unenthusiastically invited other visitors to test by saying "this is so easy";
- *efficient to use*: once the user has learnt the revealing flashlight, a high level of productivity is possible, since the comparison of the real object and its virtual counterpart takes place in the same space, and the gesture of pointing and thus switching between both is very natural;
- *easy to remember:* indeed, our experience shows that users strongly like to return using the revealing flashlight, and they immediately remembers thanks to the high affordance and the easiness of the system.
- *few errors:* with an ideal technical installation, few errors are possible with the system since it builds on an everyday pointing gesture. However, in some installations, the gesture-based optical tracking that requires somehow controlled lighting conditions was the most frequent error source.

- *pleasant to use*: we can safely affirm that the users are subjectively satisfied by using the system; they like it and are most often enthusiastic. We even observed users that laugh up loud when first understanding the system.

We have also seen in Section 1 that when studying usability, it is relevant to take into account the concept of “user experience”. In D2.1, user experience is defined according to ISO (2009) which conceives it as "a person's perceptions and responses that result from the use or anticipated use of a product, system or service".

In the V-MUXE work, based on the model of Thüring und Mahlke [Thüring and Mahlke 2007], we identified four features that affect user experience in human-technology interaction specifically for virtual museums, namely *utility*, *learnability, efficiency* and *stimulation.* Specifically for the revealing flashlight, we can draw the following conclusions : *Utility* is given since there is a natural mapping between the users' control and the system's feedback; *Learnability and Efficiency* is attained as seen above in the 5 parameters for usability and since there is a rapid feedback given by the system to users, and, of course, *Stimulation* is achieved by integration of multi-media and since the revealing flashlight is pleasant and easy to use (as also seen above in the five parameters for usability).

3.5 Portability

As seen in section 1, *portability* is a fundamental requirement for reuse of virtual museums. In V-Must, we consider two different levels of portability: the first one is an exhibition moved from one museum to another with a minimal amount of issues and difficulties in terms of transportation and adaption; the second one implies the technical concepts and services used from one exhibition re-used in an other context.

The revealing flashlight has been deployed in three different venues of the Keys2Rome exhibition (Figure 5). In one of the Key2Rome venues (the Allard Pierson Museum in Amsterdam, Figure 5 top left), we re-used the exactly same hardware as in a precedent temporary exhibition (“Eternal Egypt”, Figure 4), where the revealing flashlight was on display for more than 6 months, but with a different cultural artifact and thus a different corresponding 3D model. All installations use the same concept adapted to the different museum objects, ranging from projecting missing polychromy over digital restoration (adding details that have been eroded).

Fig. 4 - Three different setups of the revealing flashlight adapted to different museum objects and shown in three different Key2Rome venues.

Consequently, we have approved both levels of portability defined above:

- the first level of portability where we transported the same installation to different venues: the *same* cultural artifact, the *same* 3D model, the *same* tracking device and video projector, the *same* computer and software.
- the second level of portability where we adapted the concept of the revealing flashlight to different installations at different venues: Keys2Rome Amsterdam, Rome, and Alexandria. For this, we used a *different* cultural artifact, a *different* 3D model, a *different* tracking device, a *different* video projector, and a *different* computer, but the *same* software parametrized *differently*.
- an intermediate level of both levels of portability: in the Allard Pierson Museum in Amsterdam, as said above, we re-used the exactly same hardware as in a precedent temporary exhibition with a different cultural artifact and thus a different corresponding 3D model. Note that there exist various intermediate levels of portability depending on the number of similar components.

Summing up, the revealing flashlight is perfectly suited for supporting both temporary exhibitions, such as touring exhibitions, special events, expos that require infrastructures easily to carry and set up from time to time in different venues, and also permanent exhibitions. Concerning the consistent features of Portable VMs defined above, the revealing flashlight has a quite *simple setup, an adaptable design*, a *portable and durable equipment* with easily obtainable *spare equipment*.

4. Integration of VMs: 3D Assets

In this section, we present the major outcomes for an integration of enabling techniques and services for VMs. It provides efficient support for the mediation and presentation of 3D assets and is based on the developments done within V-Must. This section shall provide hints for the possibility of an easy migration and portability of assets. Nevertheless, the design of the overall frameworks deployed in V-Must took into consideration a flexible exchange mechanism.

4.1 Exchangeable Render Back-ends

This section offers an overview of the realization of flexible render back-end exchange in order to integrate multiple rendering back-ends under a common application layer for distributed systems. The primary goal was to find a practical and non-intrusive way to use potentially very different renderers in heterogeneous computing environments without impairing their strengths and without burdening the back-ends or the applications with details of the cluster environment. Our approach is based on a mediator layer that handles multi-threading, clustering, and the synchronization between the application's and the back-end's scene.

4.1.1 Requirements

In this section, we first lay down the most important requirements that guided the final design and an overview of the architecture itself. Furthermore, our approach is based on OpenSG as rendering back-end, but an easy transfer to any kind of scenegraph system providing concepts such as Fields, FieldContainers, Aspects, ChangeLists and Clustering can be done. From our experiences with past projects and the objectives of current projects, we have derived the following requirements for the extended system:

- **Extensibility and generality**. The system should be able to integrate new rendering back-ends relatively painlessly. It should also be general enough to handle back-ends coming from very different application areas and following different design paradigms.
- **Non-intrusiveness.** Neither the application layer nor the rendering back-end should need any changes or extensions in order to work together (back-ends may extend the application layer to expose specialized functionality, as described below, but basic functionality should be possible without touching both). This is important because we want to support commercial libraries as back-ends that usually come with an unalterable interface.
- **Clustering and stereo**. The system should provide (at least basic) support for rendering in a computer-cluster (tile-based and cooperative) and stereoscopic rendering, even if a back-end itself does not support it.

- **Rendering performance**. Of course, the system should allow each renderer to play out its strengths — after all, that is why we want multiple, specialized rendering back-ends. Integrating a back-end into the system should hamper its rendering performance as little as possible.
- **Efficient incremental updates**. Not only the raw rendering performance is important, but also how updates are propagated from the application layer to the rendering back-end (and sometimes the other way around). The system should provide an efficient (in terms of runtime and usability) solution to this problem.
- **Ability to extend the application layer**. While the system should rely as much as possible on a common low-level abstraction of a scene, it is sometimes practical to expose attributes that are specific to a certain back-end in the application layer. An example are extended material attributes for a ray tracer. The system should provide a mechanism to pass on such data.
- **Mixed (hybrid) rendering.** The system should provide (at least basic) support for mixing different renderers during the generation of one image. For example, it should be possible to render large static geometry with a back-end optimized for that purpose and to render dynamic 3D GUI elements in the same scene with another back-end.

4.1.2 Mediation Layer

The mediation layer is the core of the exchange mechanism. It has to be implemented for each rendering back-end (although parts can be reused). It usually consists of a specialized, in our case OpenSG-, Viewport, a scene adapter, and a context. With the Viewport, the mediator can hook itself into the rendering infrastructure (here in our example OpenSG). The scene adapter translates the OpenSG scene into the renderer's internal representation and keeps it up-to-date. The context manages instances of the back-end and allows multiple Viewports to share these instances.
Using the Viewport concept, the mediator can hook itself into OpenSG's rendering infrastructure. The scene adapter translates the OpenSG scene into the renderers internal representation and keeps it up-to-date. The context manages instances of the back-end and allows multiple Viewports to share these instances (Figure 5).

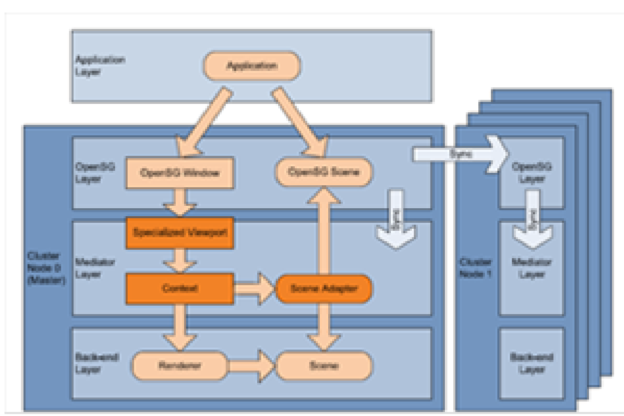

Fig. 5 - Basic design of mediation between application and render back end [Schwenk 2012]

The major advantage is that there is no direct dependency from the application layer into the mediator layer (only via the default OpenSG-Viewport interface) and no dependency of the mediator into the application layer (only into the OpenSG layer, a mediator potentially works with all OpenSG applications). Also, the mediator depends on the back-end, but not the other way around. This takes care of our requirements of extensibility and generality and non-intrusiveness. Since the back-end is fed with its own scene representation and simply asked to fill a Viewport, it usually can maintain a near-optimal rendering performance. Relying on Viewports also enables a simple (but usually sufficient) way to do mixed (hybrid) rendering. Viewports can be layered on top of each other in order to combine the images of different back-ends using z-buffering and alpha-blending.

The application layer only needs to pack data for extensions into attachments, which are then interpreted by mediators that understand the extension and are ignored by others.

Each mediator exposes a specialized OpenSG-Viewport which internally maps to the underlying renderer. So, every time OpenSG (on behalf of the application) wants a Viewport to be rendered, the back-end is invoked. The target is usually an OpenGL back buffer, but it can also be another render target. For example, one can implement a Viewport that renders an image to disk or streams a video to a website. The back-ends are invoked exclusively through this specialized Viewport class. If the application wants to use a certain back-end, it just creates the corresponding Viewport and attaches it to a window. The Viewport then creates the infrastructure necessary to convert the scene and instantiates the underlying renderer.

The scene adapter is responsible for mapping the OpenSG scene to a representation the back-end can use. This is usually where the bulk of work has to be done when implementing a new mediator layer. In some cases, the renderer may be able to use the OpenSG scene directly, or at least parts of it (e.g., an OpenGL-based deferred renderer), but usually a conversion of the scene will be necessary. In this case, the adapter will usually traverse the whole OpenSG scene graph once during initialization and build a shadow scene by converting objects such as geometries, materials, and lights into suitable representations.

4.1.3 Deployment within X3DOM

This section describes a concrete application built on the architecture above and the mediators explained above. It is a distributed visualization application for large models.

The front-end is simply a WebGL-enabled web browser, rendering a HTML5-page with X3DOM. However, current web-technology is not capable of handling large models efficiently. By “large models” we mean large for web applications, i.e. in the order of tens or hundred of millions of polygons. Therefore, we use a novel out-of-core approach to minimize the workload in the browser. The key idea is to use an asynchronous, remote culling service. Figure 6 shows the basic data flow. The browser (actually the X3DOM runtime) sends its current view frustum to the culling service, which determines the objects with the largest screen coverage and sends back a list of IDs for these objects. The browser then only fetches these “most important” objects from the asset server. This keeps memory consumption and rendering time manageable on weak devices, which would otherwise not be able to render such complex models. On the other hand, the approach consumes less bandwidth between culling-service and browser than full server-side rendering with video streams.

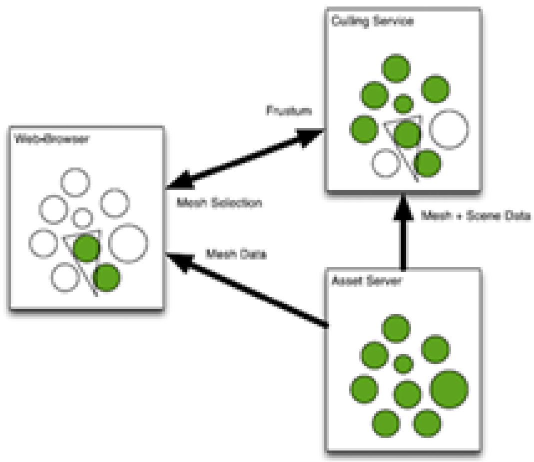

Fig. 6 - Schematic data flow between browser, culling server, and asset server

This allows us to maintain high quality and interactivity even in weaker networks, where video streaming does not work well. The culling service is an InstantReality instance running a special rendering back-end. This back-end does not render a traditional image, but calculates which objects have which coverage in the final rendering (including occlusion). From this information the sorted list of object-IDs is generated, which allows the browser to prioritize important objects. We have implemented the culling service as an Optix-based back-end (as a ray tracer) [Parker et al. 2010] and as an VGR-based back-end (as a rasterizer) [Brüderlin et al. 2012]. Both cases use a minimalistic scene adapter that basically only converts geometry and establishes a mapping of IDs to objects. The geometry conversion is shared with the other renderers in the Optix/VGR mediator. Material information (apart from transparency) is not necessary. We believe this showcase nicely demonstrates how the freedom obtained by our approach for flexible rendering back-ends can be used to build innovative distributed applications.

4.2 Code Base for Application Layer

To comply with the portability (and thus re-usability) constraints of the new presentation layer as envisaged in V-Must and its applications, several solutions require to be HTML5 conform. The code base has to be migrated or re-implemented using modern Web technologies like, CSS3, Ajax/JavaScript (JS)/Python, and WebGL. A JS library for advanced interactions provides aside basic mouse interactions also more mature possibilities such as mouse generic touch interactions providing system interaction and navigation. 3D asset deployment is plugin-free, declarative and its easy to migrate from platform to another. Major advantage is to support also web developers without a major CG background. The established framework in V-Must is JavaScript-based and allows declarative 3D graphics in HTML5, that is based on the open ISO standard X3D describing the scene and runtime behavior. The library provides several different types of nodes, each with its own function. E.g. it provides mechanism to dynamically load external .x3d objects resp. scenes stored as reference within inline nodes. A dedicated viewpoint node allows the developer to define any point of interest and related camera orientation being exposed to the user through simple markup language constructs and thus allowing to implement a guided exploration of the scene. It is easy to customize as the declarative approach provides an XML scheme wrapping 3D assets, thus capable of even providing an automatic way to deliver coordinates and orientation. In order to allow the user to switch between historical reconstructions (between two 3D scenes), the application has been realized using a pre-load option for several objects at the beginning of the render call. The management of transparency during transition uses an ad-hoc reusable JavaScript function chain. The objects are conceptually organized in two arrays of ids. This approach is quite flexible and could be applied using different parameters, but could be also a bottleneck. Here, X3DOM is capable to use also binary and compressed nodes, to increase the overall system performance and provides the above mentioned flexibility of render backbone exchange (see 4.1). The suitability of this approach has been demonstrated and showcased in different application realised within the micro projects of V-Must and during initial porting studies in Deliverable D5.3. Additionally, a very similar approach has been used for the development of the 3DHOP framework (described in detail in Deliverable 5.2 and 5.3). This has allowed a seamless integration in the CIF to build an application template for the visualization of large 3D models using a multi-res view-dependent approach (see Deliverable 5.2) and a great portability and re-usability of the applications based on this framework as discussed in Section 4.4 and Section 4.5.

4.3 Virtual Integration

In this section we outline how researchers and developers can build novel systems on top of the CIF infrastructure without being tied to a CH workflow or the web interface provided by the original CIF (Deliverable D5.2/D5.3). We introduce a way to programmatically access the powerful optimization/transcoding and delivery back-end of the CIF through a universal middleware access layer addressable by standard protocols and formats like Hyptertext Transfer Protocol (HTTP) and the JSON Data Interchange Format (JSON). To display the usefulness of our approach, we present two different cases of using the CIF pipeline as a service through the proposed access layer. We will showcase the results within an mobile iOS application that allows accessing a remote repository of 3D models. In order to interactively explore 3D models on the mobile device, a optimized version of the model is generated by the CIF pipeline, and transmitted to the mobile outlet for presentation and storage. Due to the CIF and its API no large datasets need to be exchanged between the handheld device and the repository. Second, we extend the Meshlab desktop application by a “web export" plugin. This plugin employs the CIF API to send a 3D model to the CIF which generates an HTML5 application presentation package according to some predefined templates. As a result, the users of MeshLab can easily deploy his/her work on a web site, without knowledge of WebGL or other web technologies.

4.3.1 API and Service Invocation

While the CIF allows for easy transcoding and optimization of 3D models for Cultural Heritage applications, it is inherently tied to the work flow used by that domain. With the proposed system it is not readily possible

to integrate the transcoding, optimization and templating into other applications, like batch processing scripts, native mobile applications or even yet unknown scenarios like for example an online furniture shop, where the user can customize a living room suite or cupboard and interactively try out the drawers etc. In essence, the basic steps and operations performed to transform a model into a Web usable form are generic by design. Domain-specific behaviour can be implemented through the use of highly customized output templates. In order to access the fundamental technologies underpinning the CIF, we have proposed a generic way to access the CIF subsystems by means of a standards-based API to facilitate the use of the CIF a service. (see Deliverable D5.2(b))
Here, we decided to extend the CIF using a resource-based interface. By exposing a middleware layer through a network-based service, the entire CIF pipeline, or only parts of it, can be accessed by any 3rd party application using a fixed set of API instructions. This service endpoint can be accessed utilizing the standardized HTTP protocol combined with the JSON data exchange format. A resource based architecture is employed to make full use of the HTTP protocol as well as provide a sensible future aware and easy to understand implementation. Through this API complete control of the individual steps in the processing pipeline can be offered. In contrast to other HTTP based protocols like SOAP, using elements characteristic to REST style architectures allows us to fully exploit the potential of HTTP. In SOAP for example, mainly the HTTP POST verb is used and return status codes are largely ignored. Additionally by implementing a concise JSON-based request/response content, in contrast to the verbose XML messages used in SOAP, we minimize the amount of data to be transferred over the network. This part is especially important for mobile devices which often have a limited network bandwidth capacity and/or throughput.

Calls

To recall from D5.2b, we adopted for the CIF resources through the well understood HTTP. Each resource is addressed by a unique URI (Uniform Resource Identier) combined with a HTTP verb detailing the required operation to be performed on the resource. Optionally a HTTP body containing JSON data is provided with requests that need to modify a resource. The following operations, using HTTP verbs can be performed:

- `GET` Retrieve a resource (e.g. `GET /tasks or GET /tasks/4711)`
- `POST` Create a new resource within a collection (`POST /buckets)`
- `PUT` Update a existing resource (`PUT /buckets/123/herkules.ply`)
- `DELETE` Remove a resource (`DELETE /buckets/123/herkules.ply`)

The following examples illustrate the use of our proposed API through HTTP requests. We demonstrate a typical workflow to obtain a fully optimized model, embedded in a HTML application, by providing a locally stored large model. To illustrate the working of the API on the lowest level, we make use of the cURL utitlity, a command-line HTTP client, which can be also be used in batch processing scripts.

4.3.2 Application Build

Example Mobile Repository Browser

Using mobile devices which connect to CH repositories would provide a great potential for researchers and other involved stakeholders. A device like a tablet computer could be used for efficient communication, on-site exploration, augmented reality applications, and much more. Commonly, within a 3D model archive a large number of big models are store for preser-vation. Exploring or even annotating, modifying or storing these models on a mobile device over the internet is currently not feasible. Large models take a long time to download and potentially require a massive amount of storage space a mobile device can not provide. To help overcoming those limitations, we showcase a mobile respository browser application, which utilizes the pipeline API for just-in-time transcoding and optimization of remotely stored models. For optimal performance, our iOS application makes use of a pipeline API feature which allows to convert a model stored on another server accessible by the pipeline cluster (i.e. the 3D model archive). In order to trigger the conversion of a remotely stored model, the iOS application sends the location of the large model to the pipeline servers. The pipeline servers subsequently download the remote model, optimize it for the mobile device and provide a preview as well as a downloadable bundle for the device to further process as needed

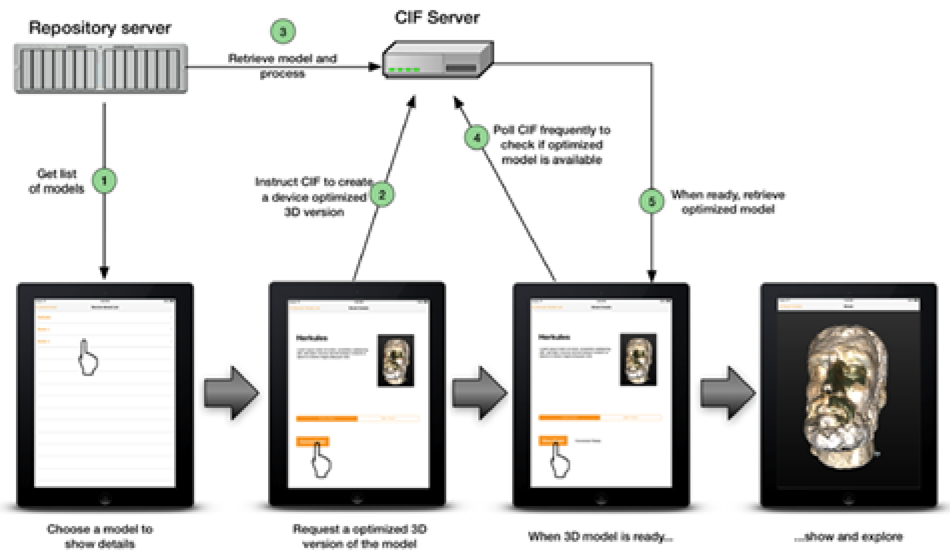

Fig. 7 - Basic operation of a mobile browser application sing the API for a virtual integration of access and transcoding.

A a list of models and metadata is retrieved from a 3D repository (1). Details are shown in a seperate view and a device optimized version can be requested through the CIF api (2). The CIF servers retrieve the requested model from the remote location and start to create an optimized version (3). The mobile device is polling the CIF servers to determine if the conversion process has finished (4). Once the model is ready, it is retrieved (5) and displayed to be explored on the mobile device.

Example Desktop extension of MeshLab

Another interesting example of use of the CIF API regards the inclusion and extension of MeshLab for exporting 3D models for the Web. This permits any user to deploy his/her content on a web site, even if the user has no knowledge of 3D graphics or web programming. A specific plugin has been designed for this purpose. The MeshLab user, after doing the desired processing on its 3D content, can use this Web Export plugin to prepare its content for the Web. The plugin requires to select the 3D model to convert (since multiple 3D models can be loaded in different layers in MeshLab), to select the application template according to which the 3D content will be prepared for the Web, and to enter a notification email to get the results, i.e. the converted model. The notification email contains also the link where to download the web package prepared by the CIF. Obviously, the selection of a different application template cause that a different web package is generated. Figure 9 shows this plugin in action. From an implementation point of view, since MeshLab uses extensively the Qt framework, the Web Export plugin simply performs the necessary HTTP requests using the Qt Network module.

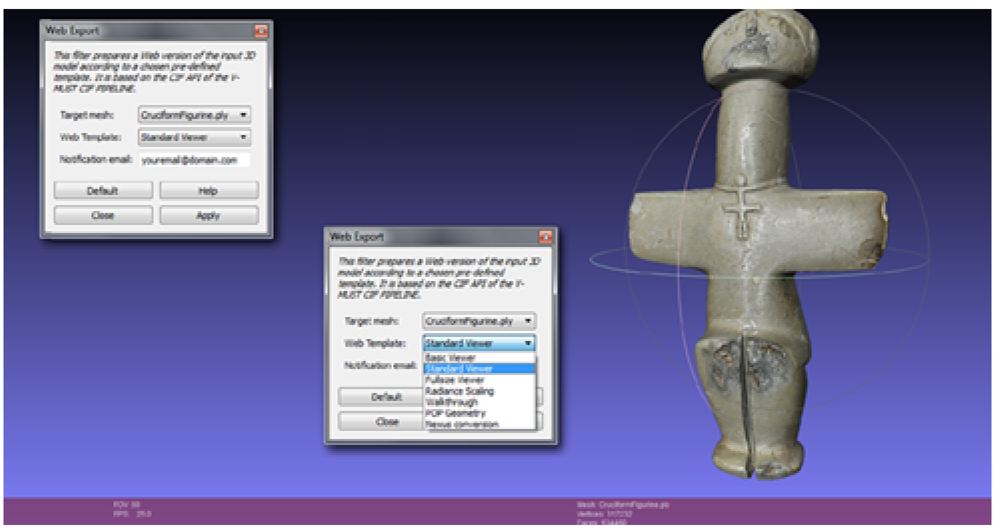

Fig. 8 - Meshlab Web Export plug-in in action.

4.4 3DHOP - Some Notes about Portability, Usability and Evaluation

3DHOP (3D Heritage Online Presenter) is an easy-to-use and easy-to-learn framework for the visualization and the presentation of 3D content on the Web. The 2nd release of the framework is available at its official web site: http://3dhop.net. Some of the characteristics of this web visualization framework making them particularly suitable to build Virtual Museum (the capability to deal with huge 3D models and customize the interaction, the capability to add hot spots to a 3D model, the possibility to drive the users to one point of view to another, and so on) but the application field is not limited to Cultural Heritage.

Since this framework has just been described in D5.2 and D5.3 we report here some considerations about the technological solutions adopted, which allows portability (see in particular the section 4.5 about this point), flexibility and usability (related to re-usability), and an (indirect) evaluation of the framework itself.

- **3DHOP is based on Javascript, WebGL, CSS and HTML5.** Since these languages are standards that a browser should follow, the portability between many browsers and many platforms is guaranteed.
- **3DHOP is designed to be used by developers with different skills.** For example, a completely naive users can copy the provided HOW-TOs and modify them to put into his/her content to publish, a Web designer that does not know anything about Computer Graphics programming can easily adapt the standard viewer to his/her needs, and a developer expert of both Computer Graphics and Web programming can easily (thanks to its modular structure) customize some of the components of the framework to built more complex web applications.
- **Many components of the framework come with safe defaults and automatic configuration** (an approach which takes inspiration from the *batteries included* one of the Python). This permits to easily re-use the code written for a specific context to anther one. This allows also the automatic generation of web pages, for example in a web service like the one described in Section 6 of the Deliverable 5.2b.

Concerning the evaluation of the 3DHOP among the community of developers in this last months we have a received a lot of feedback which allow us to assess an indirect evaluation of it. Many developers (in the order of tenths) report us to use it effectively for their aims, thus demonstrating that the rich documentation and the different level of access of the framework, allow a great number of users to adopt it (the number of download of the framework is quite high for a specific tool like this, about 370 from the 1st of October 2014 to the 31th of January 2015). 3DHOP has been employed also by the students of the course "3D Graphics for Cultural Heritage" of University of Pisa which does not have a strong programming background (the course is for student in "Informatics and Humanities"). Despite this many of them are able to use 3DHOP in

the final presentation of their (digital) Cultural Heritage projects which often consists in producing a web page about the project realized.
Another enthusiast feedback we received, which confirms the high customization capabilities of the 3DHOP, is from the *MorphoMuseum project.* A software developer inform us that he has used successfully 3DHOP to develop a big project about the publishing and sharing to the public of digital 3D models of vertebrates on the Web (see Figure QQ). This project has been realized by the Department of Paleontology of the "Institut des Sciences de l'Évolution" of Montpellier.

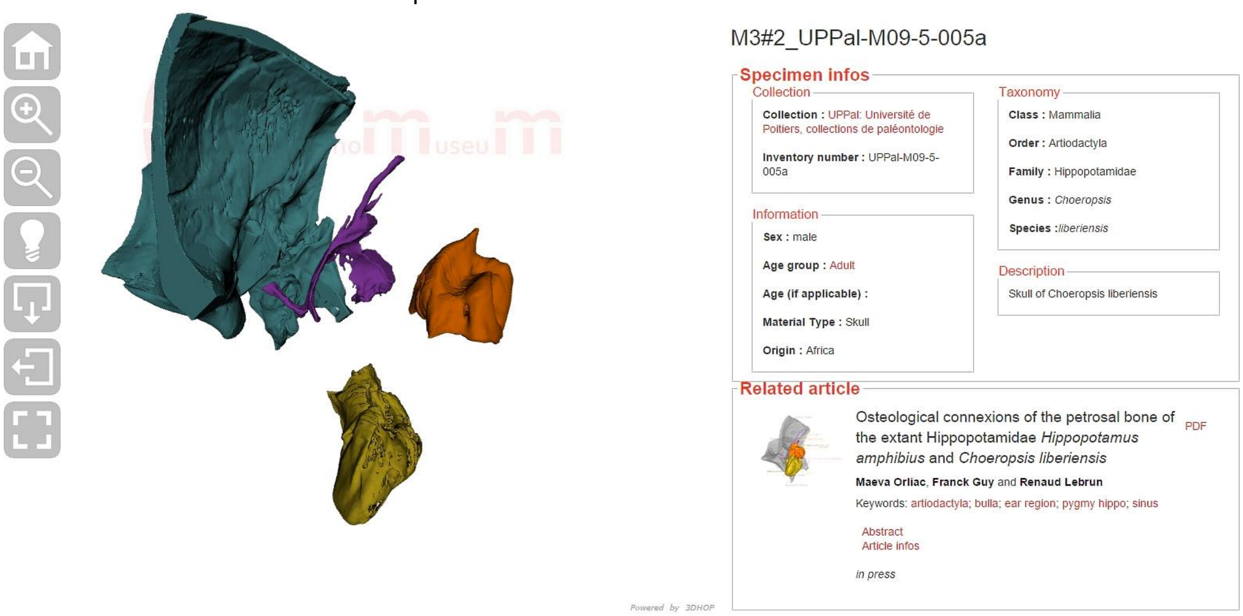

Fig. 9 - A screenshot of the MorphoMuseum project (http://morphomuseum.com). The 3DHOP viewer has been customized to visualize additional information about the species on the right and text labels on parts of the 3D model.

Concluding we can state that the technological solutions adopted and the design of the 3DHOP framework make them a good candidate to develop portable and re-usable VM. More details about portability are provided in the next section.

4.5 Portability

The realised workflow and established pipelines for service invocation and asset transformation (though not purely automatic) offers an alternative and systematic methodology to easily adopt any kind of migration.

The process actually is not fully automated, but the elaborated methodology provides a faster way to implement migrations from macro to micro VR worlds. It is especially meant for modellers and entry-level web developers. Eventually, there are no evident constraints from the original data-assets point of view, and our case study already depicted a quite complex situation. The objects generated could be quickly inserted in a web page. Using our final application scaffold as template allows even advanced functionalities, such time-shift capabilities, and viewpoints navigation to be used by web developers, as they are easily accessible through provided JS modules. We have been porting the The "Behind Livia's Villa" application to different platforms (see Deliverable D5.3) that allows the user to explore a low cost VR environment in which both the actual and the reconstructed models could be visited and switched between present and past eras. The user is guided by a collapsible map which contains a list/map of points of interest. An expert mode might allow the retrieval of meta-data information for certain objects. As a result, this demo can be interactively played in different devices which adopt different appropriate templates: desktop PC/laptop, tablets, smart-phones and as an immersive experience within Oculus Rift display (see Figure 11).

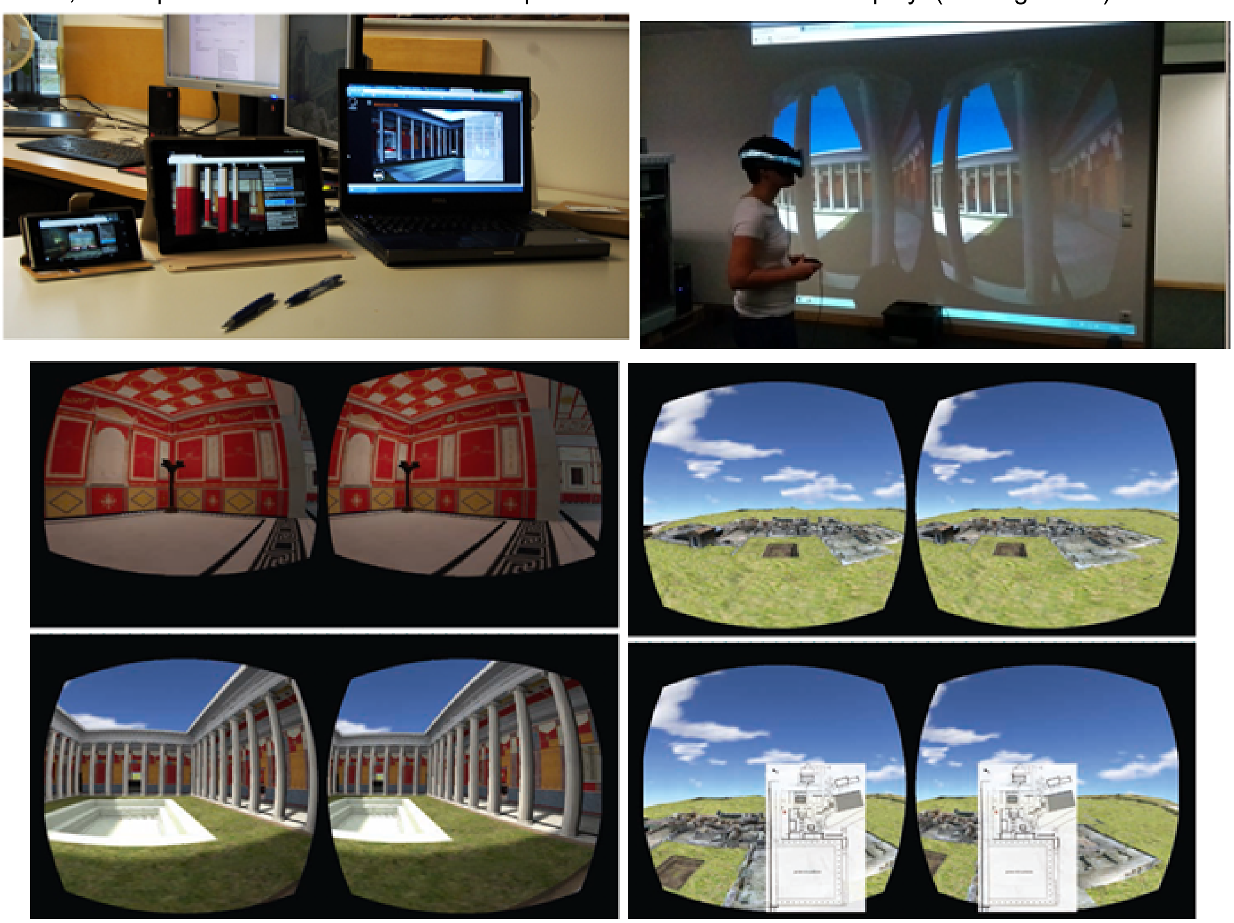

Fig. 10 - Asset deployment on different target platforms.

During the start of this migration, we performed several tests. The first attempt to convert the model and build the first version of the application, took two months including an initial learning phase. For the second release, the whole process took only few weeks. Specific “application templates” might support web developers and users alike to embed converted assets in on-line virtual museums being deployable on different devices. The results provided best practices in the creation of online services for model conversion, optimization and automatic generation of basic web 3D pages as done within the Common Implementation Framework (see Deliverable 5.2 and deliverable D5.3). An improved process and an automatic tool chain as presented in Deliverable D5.2b can be finally adopted (not exclusively) for online VMs. Any application domain relying on 3D content (e.g. automotive, aerospace, AEC) might benefit from this experience as well as the version of automatic transcoding services filling different application templates.

As mentioned in Section 4.4, thanks to the technical solutions adopted for the development, also the 3DHOP presentation and visualization framework is another successful example of a technology framework which allow the VM built on it great portability between different platforms and devices. This is possible also thanks to its *battery included* design philosophy; the components configure them automatically according to their input data and their context. In this way a web application which exploit the framework for 3D model visualization can be easily viewed in different devices, such as a smartphone or a dekstop PC. Particular emphasis has been put also on the user interaction, making possible to use the same web application on a touch screen or with a standard mouse as interactive device, with no modifications (or with very few modifications). The second release of the framework has been successfully tested on four of the major browser (Firefox, Internet Explorer, Chrome, and Opera) on different operating system (Windows 7 and 8, Mac OS, Linux) and on different devices (standard desktop PC, PC with touch screen, PC with non-standard (huge) touch screen, android tablet and smartphone and Windows 8-based smartphone). A good example which demonstrate the great portability of the 3DHOP-based application is the ***Alchemy 3D project*** (see the web site http://vcg.isti.cnr.it/alchemy for further information) developed by a collaboration between the Visual Computing Laboratory (ISTI-CNR), the Opificio delle Pietre Dure and the Peggy Guggenheim Collection regarding the three-dimensional diagnostic analysis of the painting "Alchemy" by Jackson Pollock. The aim of the diagnostic action was to create a three-dimensional high-resolution map of the geometry of the painting, usable as metric and scientific documentation, to measure, study and analyze the “materic” structure of the painting. The results of this digitization has then been used to create an interactive kiosk (using the 3DHOP framework) and a physical reproduction of the painting for the exposition "ALCHEMY BY JACKSON POLLOCK. Discovering the Artist at Work" (February 14 – April 6 2015, Peggy Guggenheim collection, Venice). The interactive installation at the exposition basically employ the same code used for the development of the viewer of the web page describing the project.

5. Conclusion

As presented all along this document, we tried to highlight the work carried out during the V-MusT.NET consortium, adopting modular strategies and transverse solutions, and being able to produce cutting-edge VMs. Specifically, we tried to produce indicators and evaluative parameters out of best practices.

As shown in Section 3 and in 4.4, we managed to produce multimedia products having in mind the necessity of being (a) long-lasting, (b) reusable, (c) portable and (d) feasible - as the same definition of "virtual museum" stresses.

We started with the definition of VM which must be usable, concurring in generating reliable approaches towards the users' experience. Furthermore, we explained how such VM need to be portable - either in terms of content portability, or infrastructure and technology. Here we underlined the tiny line which distinguishes the portability concept from transmediality and feasibility. Then, we discussed about integration and showed how cross-solutions can be suitable to produce such enhancement in the Digital Heritage domain.

The final goal of all this was to produce best practices and indications on how to create a good portable application or an effective tangible interactive application, disconnected from the physical location. Although we could not provide an effective grid of parameters, adaptable to all VMs contexts and cases, we can first say that portability, usability and integration depend on how a VM is intended according to the following categories:

1. By content
2. By interaction technology
3. By duration
4. By communication
5. By level of immersion
6. By format
7. By scope
8. By sustainability

The most interesting for us is the third category, which highly influences our principle of usability and portability. Below we provide a brief presentation:

Classification by interaction technology

This category includes all VMs defined in accordance with the type of interaction. As presented in D2.1, *interaction* is related to the user capability of modifying the environment and receiving a feedback to his/her actions. Both immersion and interaction concur to realize what is one of the main goals of a virtual experience: presence, i.e. the belief of actually being in a virtual space [Carrozzino & Bergamasco 2010]. The "Interaction Technology" category under the V-MusT.NET consortium, includes two main classifications:

- interactive VMs
- non-interactive VMs.

We obviously focused our attention on the first category. This is because Section 1.1 and Section 1.2 of this document have shown that features really affecting usability like *utility*, *learnability, efficiency* and *stimulation* increase or decrease most of the time according to the different levels of interactivity and immersivity.
An interactive Virtual Museum is indeed a Virtual Museum responding to a user's input [Oxford Dictionary 2010]. As such, it includes three interactive modes like device-based interaction, natural interaction and gesture-based interaction where users can test themselves about the system in front of them. Specifically, we have:

1. **Device-based interaction** refers to interaction with an application which is facilitated through the use of an input device, such as, but not limited to: keyboard, pointing devices (mouse), composite devices (game controllers, Wiimote), video and imaging input devices (webcam, motion sensor, scanner), and/or audio input devices (microphone).
2. **Natural Interaction** is a type of interaction that follows an approach to human-computer interaction which uses the natural behaviors (movement, gesture, speech) of the user to interact or "communicate" with an application. The interaction interface is meant to be invisible, or to become invisible through naturalized interaction (requiring limited learning for interaction), and based on nature or natural elements [Blake 2010]. Ideally, natural user interfaces are designed to use natural human behaviours to interact directly with content, and may be facilitated with devices such as multi-touch surfaces (iPad or PixelSense) or motion detection sensors (Wii or Kinect).
3. **Gesture-based Interaction** is a type of interaction that allows users to communicate with a digital system using gestures. A gesture can here be defined as any human movement that conveys meaningful information, such as waving with the hand or raising the eyebrows. With gesture-based interaction it is often easier to achieve a more natural and intuitive user interface compared to for example device-based interaction using mouse and keyboard.

All above said, we can finally affirm that usability, portability and integration cannot be generalized given their specific references to technology, interaction, context of usage and content of each VM. Nevertheless, they are not only shaped and planned during the design process of an application, but also during the testing phases of it; and finally they are compared and evaluated after its realization. Immersivity and interactivity are hence the main aspects affecting such items, which can be investigated upon the following guideline:

Usability	Portability	Integration
• *easy to learn* • *efficiency of usage* • *easiness to remember* • *reduced number of errors possible to be made*	• *simple setup* • *adaptable design* • *portable equipment* • *durable equipment* • *spare equipment*	• *usage of different application templates* • *battery included design philosophy* • *Service oriented approach* • *semi-automated strategy*

Table 2 : Guidelines for usability, portability and integration

6. References

6.1 Publications from the V-MUST.net consortium

[Gockel et al, 2013] Gockel B., Eriksson J., Graf H., Pagano A., Pescarin S., 2013. VMUXE - An Approach to User Experience Evaluation for Virtual Museums. In Proceedings “The HCI International 2013”, 21 - 26 Lu- glio 2013, Las Vegas, Nevada, USA, Ed. Springer, Heidelberg.

[Ridel et al, 2014] Brett Ridel, Patrick Reuter, Jérémy Laviole, Nicolas Mellado, Nadine Couture, Xavier Granier. The Revealing Flashlight: Interactive spatial augmented reality for detail exploration of cultural heritage artifacts. Journal on Computing and Cultural Heritage, Association for Computing Machinery (ACM), 2014, 7 (2), pp.1--18.

[Schwenk et al 2012] Schwenk, K., Voß, G., Behr, J.: A system architecture for flexible rendering back-ends in distributed virtual reality applications. In: Proceedings of Cyberworlds 2012, pp. 7–14 (2012)

6.2 Further references

[Blake 2010] Blake, J. What is the Natural User Interface (Book Excerpt). Deconstructing the NUI, 2010

[Brüderlin et al. 2007] Brüderlin, B., Heyer, M., Pfützner, S.: Interviews3D: a platform for interactive handling of massive data sets. IEEE Comput. Graph. Appl. 27, 48–59 (2007)

[Carrozzino & Bergamasco 2010] Carrozzino, M., & Bergamasco, M. (2010). Beyond virtual museums: Experiencing immersive virtual reality in real museums. Journal of Cultural Heritage, 11, 452-458.

[ISO 2009] International Organization for Standardization (2009), 9241-210.

[Nielsen, 1993] Jakob Nielsen. 1993. Usability Engineering. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Oxford Dictionary 2010] Oxford Dictionary of English, Oxford University Press, 2010

[Parker et al. 2010] Parker, S.G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., Stich, M.: Optix: a general purpose ray tracing engine. ACM Trans. Graph. 29, 66:1–66:13 (2010).

[Raskar et al. 1998] Ramesh Raskar, Greg Welch, and Henry Fuchs. 1998b. Spatially Augmented Reality. In In First IEEE Workshop on Augmented Reality (IWAR98. 11–20).

[Thüring and Mahlke 2007] Thüring, M., Mahlke, S.: Usability, aesthetics, and emotion in human-technology interaction. In: International Journal of Psychology. vol. 42, pp. 253-264. Psychology Press, London (2007)

[Wu 2010] Wu, A., Tangible Visualization, ACM Press, Proceedings of Tangible and Embedded Interaction 2010 (TEI '10), p. 317 - 318, 201Carrozzino, M., & Bergamasco, M. (2010). Beyond virtual museums: Expeencing immersive virtual reality in real museums. Journal of Cultural Heritage, 11, 452-458.0.