Analysis of Visualisation and Interaction Tools Authors

Massimiliano Corsini, Roberto Scopigno, Luigi Calori, Holger Graf, Daniel Pletinckx, Sofia Pescarin, Guido Lucci Baldassari, Emanuel Demetrescu, Daniele Ferdani, Patrick Reuter, et al.

▶ **To cite this version:**

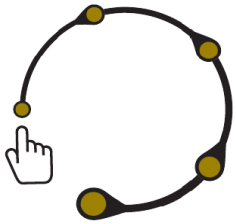
Massimiliano Corsini, Roberto Scopigno, Luigi Calori, Holger Graf, Daniel Pletinckx, et al.. Analysis of Visualisation and Interaction Tools Authors. [Research Report] V-MuST.net. 2012. hal-02320960

HAL Id: hal-02320960
https://hal.science/hal-02320960v1

Submitted on 19 Oct 2019

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

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



EXPERIENCE THE FUTURE OF THE PAST

DELIVERABLE REPORT

Document identifier:	**V-MUST.net - D 5.1**
Due Date of Delivery to EC	**End of M12**
Actual Date of Delivery to EC	**16/03/12**
Document date:	**05/11/11**
Deliverable Title:	**Analysis of Visualisation and Interaction Tools**
Work package:	**WP 5**
Lead Beneficiary:	CNR
Other Beneficiaries	CNR, FHG, CULTNAT, KCL, ULUND, INRIA, FHW, VISDIM
Authors:	Massimiliano Corsini, Roberto Scopigno, Luigi Calori, Holger Graf, Daniel Pletinckx, Sofia Pescarin, Guido Lucci Baldassari, Emanuel Demetrescu, Daniele Ferdani, Patrick Reuter, Xavier Granier, Mattias Wallergård, Joakim Eriksson, Drew Baker, Hugh Denard.
Document status:	**Final**
Document link:	http://www.V-MUST.net/media_pubs/documents

	DELIVERABLE REPORT	*Doc. Identifier:* **D. 5.1**
		Date: ***16*-03-2012**

Changes

Version	Date	Comment	Authors
1.1	03/11/2011	**First version** Layout definition and table of contents	Holger Graf, Sofia Pescarin
1.2	15/01/2012	**Second version** Workflow analysis, tools analysis, UI Design	Sofia Pescarin, Mohamed Nabil, Guido Baldassari, Emanuel Demetrescu, Anthousis Andreadis, Massimiliano Corsini, Mattias Wallergard, Xavier Granier, Holger Graf
1.3	31/01/2012	**Third version** Contribution to the tools section, first version of integration mechanism, Design of the CIF	Massimiliano Corsini, Xavier Granier, Patrick Reuter, Holger Graf
1.4	20/02/2012	**Fourth Version** 2nd version on Design of the CIF, Online museums, workflows and 2nd version of integration mechanism CIF	Massimiliano Corsini, Xavier Granier, Holger Graf

Copyright notice:
Copyright © V-MUST.net.
For more information on V-MUST.net, its partners and contributors please see http://www.V-MUST.net/
The information contained in this document reflects only the author's views and the Community is not liable for any use that may be made of the information contained therein.

Table of content

Executive Summary

This document provides an in-depth analysis of visualization and interaction tools employed in the context of Virtual Museum. This analysis is required to identify and design the tools and the different components that will be part of the *Common Implementation Framework (CIF)*. The CIF will be the base of the web-based services and tools to support the development of Virtual Museums with particular attention to online Virtual Museum. The main goal is to provide to the stakeholders and developers an useful platform to support and help them in the development of their projects, despite the nature of the project itself.

The design of the Common Implementation Framework (CIF) is based on an analysis of the typical workflow of the V-MUST partners and their perceived limitations of current technologies. This document is based also on the results of the V-MUST technical questionnaire (presented in the Deliverable 4.1). Based on these two source of information, we have selected some important tools (mainly visualization tools) and services and we elaborate some first guidelines and ideas for the design and development of the CIF, that shall provide a technological foundation for the V-MUST Platform, together with the V-MUST repository/repositories and the additional services defined in the WP4.

Two state of the art reports, one about user interface design and another one about visualization technologies have been also provided in this document.

1 Introduction

This document presents the *analysis of the visualization and interaction tools*. This analysis is required to identify and design the tools and the different components that will be part of the Common Implementation Framework (CIF). The CIF will be the base of the web-based services and tools to support the development of Virtual Museums with particular attention to online Virtual Museum. The main goal is to provide to the stakeholders and developers an useful platform to support and help them in the development of their projects, despite the nature of the project itself.

In the next section, we provide a detailed description of the workflows of the V-MUST partners in order to assess better the needs of the CIF, and hence the corresponding tools/services should respond. After this, we briefly describe the proposed tools and services. Due to the aim of this document we focus more on visualization tools, user interfaces and services where visualization plays an important role. Other functionalities of the V-MUST platform, like the repository for archiving and searching, for example, is described in the document "Deliverable 4.2" that is tightly related to this document.

2 Description of the Workflows

This section presents several real workflows, which represents the practical experiences of some V-MUST partners. Our goal is to start from the description of those experiences and analyze them in order to assess and generalize the needs and requirements of the services/tools that should be taken into account for the design of the components of the Common Implementation Framework. The workflows are presented in a standardized way to facilitate the analysis and the comparison between them. A detailed, less uniform, description of such workflows is provided in the Deliverable 4.2. In section 5 we provide a description of the tools/components that fulfill the needs and requirements coming from the analysis of these workflows.

2.1 Real-time 3D on line and off-line production at CNR-ITABC

The Virtual Heritage Lab of CNR ITABC (www.vhlab.itabc.cnr.it) has been involved in the creation of several different types of Virtual Museums:

a) off line museum installations (Movies, Desktop-based VR or Natural Interaction VR)

b) on line applications (real time exploration of 3D landscapes/sites)
c) mobile devices such as tablets/smart phones (Multimedia projects, VR exploration of reconstructed sites)

2.1.1 Concept Creation

The usual approach is to start with the definition of the **concept** (i.e. define the final results and the raw data to be built to reach the goal); then, this phase produce a document to present the concept, the story we want to tell. Tools for storyboard authoring could leverage and help considerably this step of the work. The following step is to plan the **narratives**, starting from historical sources and other creative work. At this stage texts and audios are produced (that at the end will be integrated in the application).

2.1.2 Digital Assets Creation

An extensive multimedia data acquisition activity is usually planned, including GIS data, 3D acquisition or modeling, acquisition of photos and videos. Two types of workflows, depending upon the type of data needed for the specific installation or communication product, are typically adopted:

- *Sampled Models.* In this case, 3D data are acquired by directly sampling reality, acquiring data on the field with Image-Based or Range-Based techniques (3D scanning); those 3D models are usually integrated with GIS-based datasets.
- *Reconstructed Models.* In this case digital 3D representations are produced by means of a standard modelling software or by using procedural techniques; this allows to build a reconstruction of a site or of an architecture as it was in a certain historical period.

In both cases the following digital assets are produced:

- *Raw Data (acquired data).* Data formats commonly used are: OBJ, PLY, XYZ, SHP, DXF, GEOTIFF, TIFF, DEM;
- *High Resolution Models*, encoded by producing geometry plus texture(s), used mainly for 3D rendering and for CG animation. Data formats commonly used are: OBJ, MAX, BLEND, TIFF, OSG terrain;
- *Low Resolution Models,* i.e. models optimized for real time rendering purposes by using geometric simplification technologies. Data formats commonly used are: OBJ, MAX, BLEND, OSG, JPG.

In each step, the different professionals involved in the design and implementation of the communication project would find extremely useful the possibility of using an enhanced repository for digital assets, providing

functionalities for uploading and downloading data together with metadata and, after further processing, being able to uploading it again with some information on the modification introduced.

2.1.3 Concatenation of Data into story

Data are connected through a specific story using an authoring environment that is in charge of the creation of the final application (according to the three types of applications described in section 2.1). ITABC-CNR work team do not have a specific competence regarding user interface design and do not have a structured repository for digital assets. The communication among the design and implementation staff is based on emails. FTP and SFTP with no explicit metadata management is also employed to exchange the digital assets. The documentation part is often realized with the non-commercial tool *Google Doc* for allowing cooperative work on textual documents.

2.1.4 Presentation and Interaction

The final goal of all our effort in the design of content for virtual museum is to provide visual experiences for either specialists or for the general audience. Visualization techniques allows to present and illustrate important information either contained or abstracted from the digital assets. Similarly to scientific illustration, this may inspire or help story telling. Interaction techniques allow expert and general audience to explore the digital content and for general audience to be part of the story.

2.2 On line virtual museums creation at University of Sarajevo

The *Sarajevo Graphics Group* at the Faculty of Electrical Engineering, University of Sarajevo is working on two types of virtual cultural heritage applications:

1. *Virtual reconstructions of cultural heritage objects* (either because damaged or destroyed); an example is the Virtual Reconstruction of Isa Bey Tekija (see: www.muzejsarajeva.ba/tekija*)*
2. *Virtual museums*

2.2.1 Concept Creation

The communication with the user is at the moment quite simple; it is based on a simple Web interface and a VRML browser. In the near future the aim is to adopt for more sophisticated interactive storytelling and

immersive technologies. The use of immersive technologies will require to modify the virtual environments currently developed to make it more suitable for immersion.

2.2.2 Digital Assets Creation

The ***virtual reconstructions*** of cultural heritage objects is currently based on the following workflow:

Data collection => 3D modelling => Export to web 3D => Digital storytelling =>
=> Web design => Web implementation

Data collection. At the beginning of the process all available data about the selected objects are collected (historical background, blueprints, archive photos, photos of the remains) in collaboration with historians, archaeologists and representatives of cultural heritage protection institutions.

3D modelling. The 3D model of a real object of interest is created according to its final use. If the final use of the object is to be part of a web-based application, simple geometry is created. Sometimes, first a high quality version of the model is created and then this version is optimized to fulfil the constraints of the various applications. After creating the geometry of the object,
textures are added to improve the visual aspect and realism. We use as much as possible textures originated by real photos of object's remains. If the object does not exist any more, the textures are created in collaboration with experts, archaeologists or historians.
Defining a proper illumination of the virtual object is the next step in the process. The lighting environment is chosen to provide a satisfactory level of realism to the model and whose effect should survive the export of the scene in web 3D.
Finally, we create pre-defined views on the model, that will become viewpoints after exporting the data in web 3D.

Export in web 3D. In the case of the *Isa bey tekija* installation, the model is exported in VRML, in order to support online visualization using standard VRML browsers (Cortona, Octaga etc.). Being not completely satisfied with the result of VRML exports, some of the objects created have been exported in Flash using panoramic renders with hot spots (e.g. the virtual presentation of the *Church of the Holy Trinity* in Mostar).

Design of Virtual Museums

Having very limited funding to invest in the design and implementation of virtual museum projects, we usually created them by the help of students and by proposing them to collaborate to laboratory projects as a part of the assignement of our Computer Graphics course. The work for developing a Virtual Museum is typically organized according to the following workflow:

Selecting exhibits => Photographing => 3D modeling => Digital Storytelling =>
Creating Virtual Museum structure => Design and implementation of the digital content

Selecting exhibits. In collaboration with museum curators we usually select as many exhibits as we have a sufficient number of students which could help in the production of the virtual museum. Every student would be in charge of one exhibit. Curators would provide the background materials on each exhibit, such as short story about its origin and purpose.

Photographing. Students would take photos of objects for the photo gallery joined to each object's model and for the textures.

3D modeling. Using the modeling techniques they learn during the course, students would create realistic 3D models of objects and export them to VRML. In one of the projects students were also using photogrammetry improved with 3D modeling (digital catalogue of stecaks).

2.2.3 Concatenation of Data into a story

Digital storytelling I. Digital stories for our applications are created with the following pipeline:

Scenario => Storyboard => Collecting materials => Recording narration =>
=> Editing => Postproduction

After the initial scenario is created, we draw the storyboard of the key moments of the story. Then we collect or produce material from the storyboard (sometimes we use animation or renders of the 3D model, animated guides or live actors recorded against a green screen background). We select or compose the music and record narration. Then, we edit all material according to the rules of film language grammar. The digital story is finalized through sound and video postproduction.

Digital storytelling II._In order to provide the user with the context of the object, its purpose, and other details that cannot be discovered through browsing the model, the students created digital stories using the material provided by curators and other materials they could find from available sources.

2.2.4 Visual Presentation and Interaction

Case 1 - Web implementation_We design and implement a web site for each artwork, that contains texts about the object, photos of the present appearance and an interactive 3D model. Some of the objects have a digital story linked to the web site and some have it incorporated in the virtual environment.

Case 2 - Creating Virtual Museum structure_The final shape of the_Virtual Museum structure is still a subject of our research. By now we implemented two kinds of structures:

1. ***Central 3D virtual environment,*** with proxies of objects that are linked with each object's web site. Object's web site contain text about the object, gallery of photos, digital story and interactive 3D model in VRML (an example is the Virtual Museum of Bosnian Traditional Objects, available at http://www.muzejsarajeva.ba/btp).
2. ***Story guided virtual environments*** This structure has a central digital story explaning the context of the exhibition. The story stops and the user can browse gallery of objects mentioned in the story. After the user finishes visiting objects (their individual web sites organized as in the previous example), the story continues till the next gallery. This concept is implemented in the Sarajevo Survival Tools project (available at the web address: http://h.etf.unsa.ba/srp/)

Design and implementation of digital content
After the Virtual Museum structure is defined we create web design and implement the content accordingly. Some web sites are implemented in html and some in Flash.

Tools used in the different phases of the work :

- *3D modelling* - 3ds max, Maya, PhotoModeler
- *Digital storytelling* - Adobe Premiere Pro, Adobe After Effects
- *Web design and implementation* - Dreamveawer, Flash
- *Web 3D* - VRML, Flash, x3D

2.3 Virtual Reality production at the Foundation of the Hellenic World

The creation of a Virtual Reality production (see Figure 2.1 for an example) includes a lot of work from different people that needs to be organized, checked and matched with time and budget constraints. The main workflow and practices used in FHW are described below. Let we introduce here a couple of basic issues.

Figure 2.1 A screenshot from *A Journey to Ancient Miletus.* Virtual reality reconstruction about the city of Miletus shown at the Virtual Theater "Tolos".

Criteria to decide upon interaction and visualization modes

The *mode of interaction* depends on the scope of the virtual museum (i.e the prospected audience, a single user *vs.* a small group 5-10 people *vs.* a large audience). The *visualization mode* is also decided upon the end user, e.g. if the production will be hosted as a portable installation or on a large screen facility. We need to clarify that points in the early stage of the design of a new production. When have to focus on a small group installation, we give more emphasis on the interactivity of the production (i.e camera tracking etc), while on the other hand when we have to address large group installation the emphasis is given on the visualization aspect of the production (e.g. enhanced 3D graphics).

Bringing a museum to the web - wish list

To bring a Virtual Museum on the Web, one of the pressing need is the availability of a visual tool that will allow the user to easily set his production. More specifically, a tool that will provide not only intuitive, yet precise, 3D asset placement, lighting of the scene, embedding interactions but also facilitate the construction of

a scenario (e.g. an authoring environment for organizing spatially the 3D content and posting it of Virtual Museum on the web, a preview mode for web display, etc).
Additionally, a web viewer should support some sort of advanced visualization techniques in order to be able to visualize the content a accurate as possible.

2.3.1 Concept Creation

Plan the activities and the workflow

In this phase we need to identify end-user specifications of the interactive installation. Is it going to be guided by a human navigator? Is it going to be a dome-based production or a Reality Center-type or production? These decisions define different pipelines to be followed for the content creation.

Gather reference material and lay down the project into more manageable units (scenes) and storyboard design

Based on the concept of the production, an initial scenario (text) is created. At this step all the reference material is created/gathered and it will be later used by the modelers and texture artists in order to achieve the desired historic representation. Depending on the production needs, the scenario is either composed of small blocks (imagine them as small spheres, self-contained) that we move from one to another through some special effect process, or a complete big world that contains the whole assets. In this process we create also a sequence of sketches matching each scene. These sketches identify/describe the aesthetic of the production as well as the main points of interest, that each specific scene will address.

2.3.2 Digital Assets Creation

Audio Recordings

According to the scenario, dialogs and music may be required. In this phase we create the scripts that describe the interaction and in general the character activity in the production, and send them to studio for recordings. The timing information from the sounds will be used later for the creation of the characters animation.

List the foreseen needed assets to be developed and plan a comprehensive data organization

For each defined scene a list of major assets is formulated (main buildings or areas that will be virtually visited). Furthermore, smaller assets that will be manipulated during the process need also to be identified (such as characters, cloths, animals, small filler items, etc.).

Organize the asset production into a repository allowing to share work among the work group

We have common server folder structure that identifies how the asset (model - textures - code) is exchanged. For each asset a folder tree is created storing

- reference material
- first draft mode
- high resolution textures
- real time model and real time textures

One person is responsible for each folder as we do not use any conflict-resolving tool. Architects and historians fill the reference material and fist draft model. The texture artist produces the high resolution textures and a modeler constructs the real-time model and assigns the real-time textures.

Modeling - Texture Creation - Character Creation and Animation

At this step high resolution geometry and real time geometry are created for all assets and characters, based on the reference material gathered. The same goes for the textures that will be used. At this stage also the character rigging and animating takes place.

Tools Used

- *Buildings - Assets Modelling:*
  - Softimage XSI
- *Trees and Plants:*
  - SpeedTree Modeler
- *Character Modeling - Rigging - Animation:*
  - 3D Studio Max
- *Character Texture Creation:*
  - Unfold 3D (Used for UV unwrapping)
  - BodyPaint 3D (Used for characters textures painting)
  - Zbrush (Used for characters normal maps creation)
- *Cloth Creation:*
  - CLO3D (Used for characters clothes creation)
- *Texture Creation:*
  - Adobe Photoshop
  - PixPlant plugin (normal maps)

Final Production Formats

- *Buildings - Assets:*
  - .osg (OpenSceneGraph ascii file format)
  - .ive (OpenSceneGraph binary file format)
- *Trees & Plants:*
  - .srt (Speedtree placement file format)
  - .stf (Speedtree model binary file format)
- *Characters & Animations:*
  - .cfg (cal3d character-animation file format)
  - .csf (cal3d skeleton file format)
  - .caf (cal3d animation file format)
  - .cmf (cal3d mesh file)
  - .crf (cal3d material file)
  - .fbx (Autodesk file format)
- *Textures:*
  - .sgi

Conversion Tools

- *Buildings - Assets:*
  - XSI to OSG converter (in house developed)
- *Characters and Animation:*
  - Cal3D Plugin for 3D Studio Max
  - FBX plugin for 3D Studio Max
- *Textures:*
  - Image Library (in house developed). Supports bmp, png, tiff, tga, jpg, sgi, gif

2.3.3 Concatenation of data into story

Conversion to engine formats - supervising and monitoring the asset quality and consistency - organizing workflow activity into controlled steps

The models-textures-animations are processed and rendered in our rendering engine. At this step the real-time materials are created and assigned, while each asset is checked and revised many times to assure the desired quality.

Scenario Implementation - Production Synthesis

The assets are placed in the world and the described scenario events and interactions are created at this step using the scripting languages supported by our rendering engine. Sounds are also imported at this step in our productions.

2.3.4 Visual Presentation and Interaction

The models-textures-animations are processed and rendered in a proprietary engine, developed by the Foundation of Hellenic World. As previously stated, interaction devices and modalities depends on the scope of the virtual museum and by the user-modality, i.e. single user, groups of users and so on. (i.e the prospected audience, a single user *vs.* a small group 5-10 people *vs.* a large audience). According to the number of people use the Virtual Museum installation the emphasis is moved from the visualization aspects (large audience) to the interactivity (small group of users-single user).

2.4 Culturama at Cultnat

2.4.1 Concept Creation

We describe the workflow at CULNAT by presenting a specific example among our productions, whose goal is to enable the virtual visit to the *temple of Horus at Edfu*, Egypt.

The system used for presentation to the public, CULTURAMA 2, in its current status is composed of a half-circular display that is made up of 9 adjacent screens, each one 120 cm in width and 270 cm in height. The angle covered by the union of all the screens is about 160 degrees. The complete projected image (1080 cm x 270 cm) and the curvature of the display increase the feeling of immersion, especially when observers are seating near the sweet spot (geometrical center of the screens circle). The projected image is composed by 3 different HD projectors; each one projects a pre-distorted image on 3 screens. The reason behind using flat screens is to have backward compatibility with a previous system, CULTURAMA 1. The 3 HD projectors are connected to an external GPU from NVIDIA, QUADRO PLEX.

The application running on the system is developed using the VIRTOOLS software. The application enables the virtual visit in 2 modes:

1) automatic camera navigation on a pre-defined paths;
2) interactive mode based on user's input.

Both modes run in real-time and users can simply switch between them at any time.

The system in its current status is only the first milestone in a long-term plan to develop a global Virtual Museum presenting the Egyptian Civilization. Some of the planned features to be added are:

- better interaction with the content using touch-based intrfaces or natural interaction modalities;
- enabling stereoscopic vision.

From the content point of view, the temple visit should be linked to the Egyptian Civilization timeline (that is a parallel project) and more sites have to be added and accessed from the timeline interface .

2.4.2 Digital Assets Creation

The 3D model of the temple was built using the commecial 3D Studio MAX modeling software, using as a reference some old drawings of the temple. The temple walls were photographed using a digital camera and Hugin panorama software was used to stitch walls photos into one big photo mosaic. The photo mosaic is then assigned as texture to the 3D model enhancing the realism of the model. Before being imported into VIRTOOLS, the temple model has been processed by a script written in 3D MAX to optimize it for real-time environment. This script does mainly 3 tasks:

1. bake lights into texture;
2. optimize the meshes to run into VIRTOOLS (for example it changes the mesh channels as required by the VIRTOOLS software);
3. export each part of the model into separate VIRTOOLS files that can be imported into the application dynamically, improving the final overall performance.

The separation of models into files has the advantage of leaving the control of what has to be loaded and when to the application; this is important when running the application on different systems and needs to optimize the loaded model according to the system capacity.

2.4.3 Visual Presentation and Interaction

We described here the usual phases to setup our visualization multi-display system:

Load: In this step we control the 3D data loading according to system capabilities.

View: In this step we decide the current display parameters (screen size, angle between screens, projectors positions…), which depend on the specific hardware installation.

Navigate: In this step we decide the navigation method that we want to apply for the specific Virtual Museum application. Generally, it is a mix between pre-designed animations and free camera navigation.

Setup: This is the final display stage. In this step we build the setup according to space constrains and clients requirements.

Tools used in the different phases of the work:

- *Data preparation:* VIRTOOLS>VSL, MAX script
- *Virtual Museum Authoring:* VIRTOOLS> VSL
- *Setup:* VIRTOOLS>VR player

2.5 Computer Graphics Stereo Production at CINECA

Cineca has developed a personalized approach to the diffusion of knowledge, both scientific and humanistic, encompassing ancient and contemporary history up to art. Repositories, databases and virtual reconstructions have been at the core of several projects; experiences converged to our recent stereo animation productions. The rest of the description is therefore mostly oriented to the description of our workflow for the production of computer-animated videos. Our projects always aim at using open-source tools, rigorous scientific basis, trans-media broadcasting and, increasingly, an effective story telling framework. For example the 3D stereo show on the evolution of the Universe from the initial Big-Bang to the formation of galaxies and stars, realized for Turin and Chicago Planetarium, is based on animations produced by means of computing-intensive scientific simulations.

For cultural productions (see Figure 2.2) it is important to define a clear concept and to set up an interdisciplinary team with all the different professional skills, in accordance to the media involved. We consider also mandatory to use a repository to allow the versioning of the storage for most of the data involved in the

workflow. A repository engine (SVN) has been installed within the Cineca HPC cluster to allow to steer the rendering tasks in a data-driven manner, where the evolving content will drive the request for new rendering.
Most of the interactive tools (Blender, Meshlab, Cityengine) are installed on remote visualization nodes, to allow remote interactive handling of large data models requiring large memory and fast CPUs.
The traditional 3D Blender production pipeline has been widened with specific features and constraints, enabling the philological approach, so important in our vision, to be integrated within the creative process.
The pipeline must support GIS data, laser scanned data, landscape reconstruction and three-dimensional models and simulations coming from previous research projects. Whenever possible, an open source approach is followed to ensure a future use of the models and to allow the creation of scripts to better manage and control data and processes.
3D HD movies are rendered on the Cineca's Blender Render Farm, which differs from normal Blender rendering farms since it runs on a high-performance Cineca HPC cluster.
Moreover, projects are always seen as a training opportunity and an open content resource for future use. The open content produced within stereoscopic movies should be available for new projects; we encourage the reuse on various platforms, including mobile devices, 3D digital TV programs, serious games, etc. In order to widen the accessibility we are also designing natural interface modules to navigate virtual heritage environments.
In the following the presentation is structured in a quite different way with respect to the other workflows to underline better some aspects of our production. The phases here described in detail are the Pre-production, Production and Post-Production phase.

Figure 2.2: A frame of the animation movie "APA: The Etruscan" realized by CINECA.

2.5.1 Pre-production

Creative perspective

The director provides the **Screenplay**, a document which narrate the story scene by scene, detailing the *locations* and the *actions*, *expression*, and *dialogues* of the characters. From this document we derive the **Breakdown** which further split the scene in **Shots** and, for each shot, details the position and movement of the camera. The text of the dialogue is nailed down and provide the shot time length. Further indication in this document can be the time of day, the type of the lighting, the music and the environment sound.

For each shot one or more images are created in order to have a first visual representation of the movie; these images will constitute the initial **Storyboard**. The previous aforementioned documents will drive the actors while recording each character **speech line**. Finally, by putting together the Storyboard and the recorded speech we create the **Animatic**, which is a draft version of the movie with a very poor visual but with the correct timing. This document allows the Director to review many aspect of the movie, such as the rhythm of the narration, and to take strategic decisions in advance. When the Animatic is approved, the sequence of scenes and shots is fixed.

Technical Perspective

In a digital movie all the different activities produce digital contents, often referred to as **Assets**. Assets can be transmitted trough the Internet, allowing people to do workgroup even if they are displaced in different geographical locations.

In general it is not possible to merge two different versions of an asset, so care should be taken to avoid having two artist working on the same asset at the same time, in short *assets should be edited in mutual exclusion.*

To hold all the production data we set up three different *online repositories*:

- **unversioned repository**: used to hold all static data and reference materials;
- **versioned repository** (svn): used to hold all the movie Assets;
- **manually-versioned repository**: used to hold the frames generated by the render farm(the intermediate versions of the movie).

The difference among the last two repositories is that the svn preserves the history of all data, even the deleted data, while the other keeps only the latest two or three version of the data. Other technical step are the Render farm Web Interface, and the Production Blog.

Management perspective

The production of a movie (even a short one) is a huge work. Many people must be involved to keep the time of its development within reasonable time limits, often employed by different companies. The production should enable them to work in parallel, not hindering each other. So it is important to understand which stages of the work are inherently serial and which ones can go in parallel. Hence it is possible to deliver a first production plan and to foresee a time-frame for hiring the different teams. The following picture (Figure 2.3) show a possibly subdivision in stages.

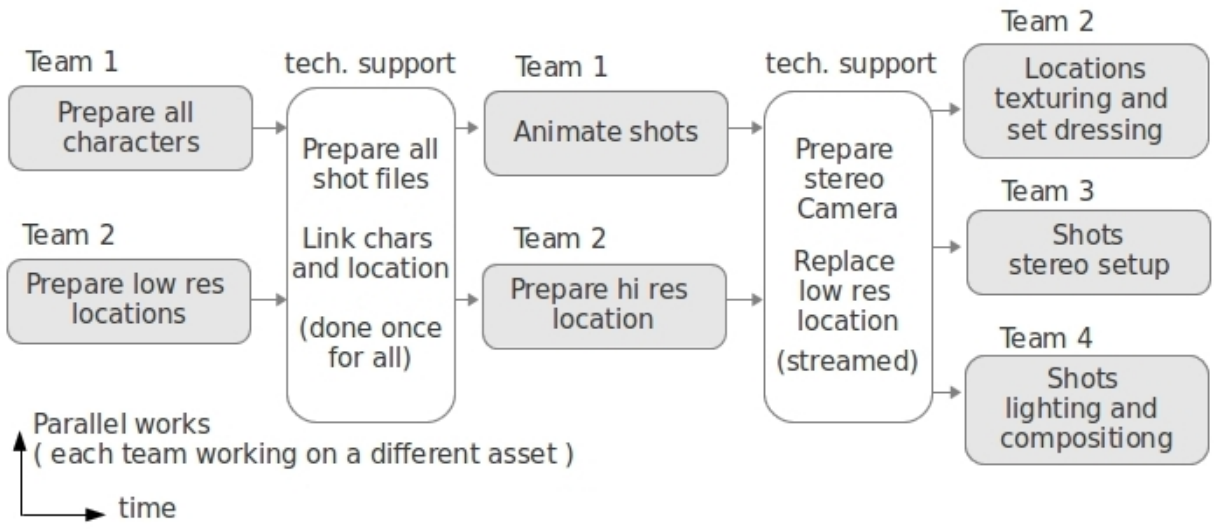


Figure 2.3 Stage subdivision of the production of a movie.

Asset Files and Repository organization

Given the need to access the assets in mutual exclusion, we can improve Artist's parallelism by having assets represented by a large number of small files instead of just a few big ones. To achieve this aim we largely exploit the Blender capability to create links to external files.

Figure 2.4 shows the general structure of a shot file, which embed the camera and character animation, the lighting and the compositing network, but links the models of character and locations. Other files may embed models or recursively link to sub-models; at the end of the chain there are the textures files.

The production repository is divided in scene and library. In the scene branch there are all the shots files organized by scene. In the Library branch we have: the locations, the characters, props, and all the textures. Shot-files always link assets in the Library, files in the library always link files in their subfolders. All the links are specified as relative-paths.

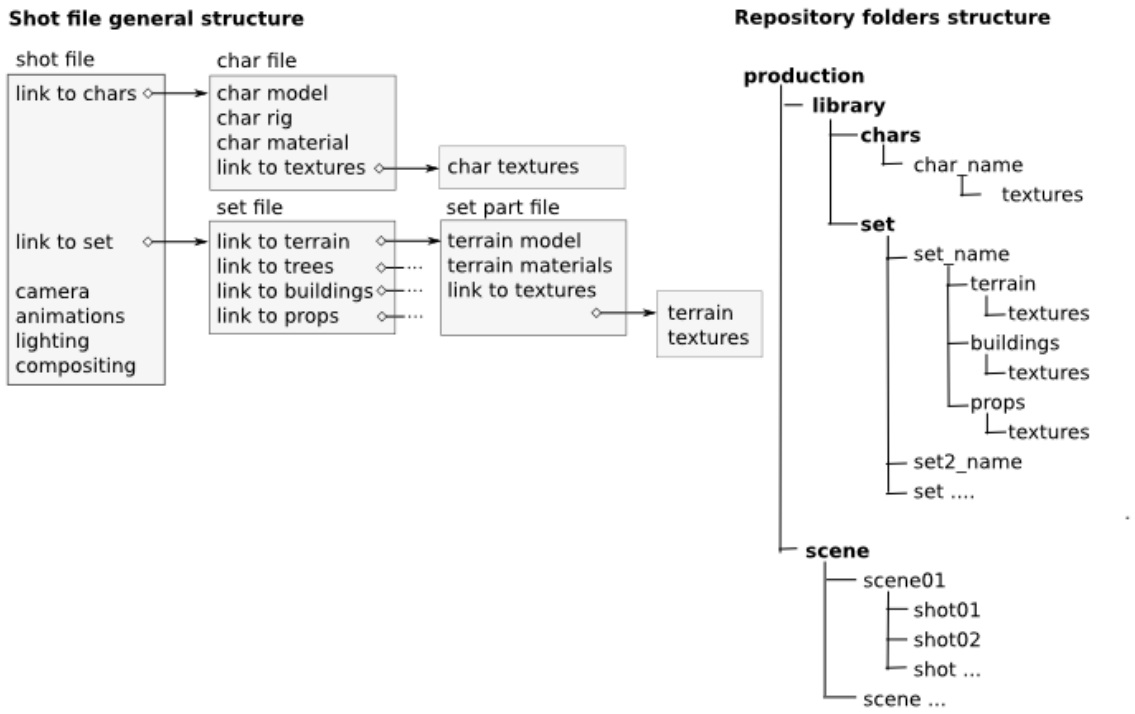


Figure 2.4 General structure of a shot file.

This overall organization is inherited from the project Big Bucks Bunny, and we must here acknowledge the Blender Foundation for having released their production files in public domain, we also acknowledge Spark Digital Entertainment for sharing their knowledge in production management.

Using external links, Artists are required to have on their own machines the Asset being manufactured but also all of its dependencies. To do this we use a feature of the versioning-system, that is the ability to simply

recreate on the user machine a copy of the whole repository (**checkout**) or a part of it. By using the politics to specify the external reference with relative path, the Artist is free to checkout the repository where he likes.

2.5.2 Production

The production is the joint work of Artists, Technical people and Supervisors; its workflow is shown in Figure 2.5.

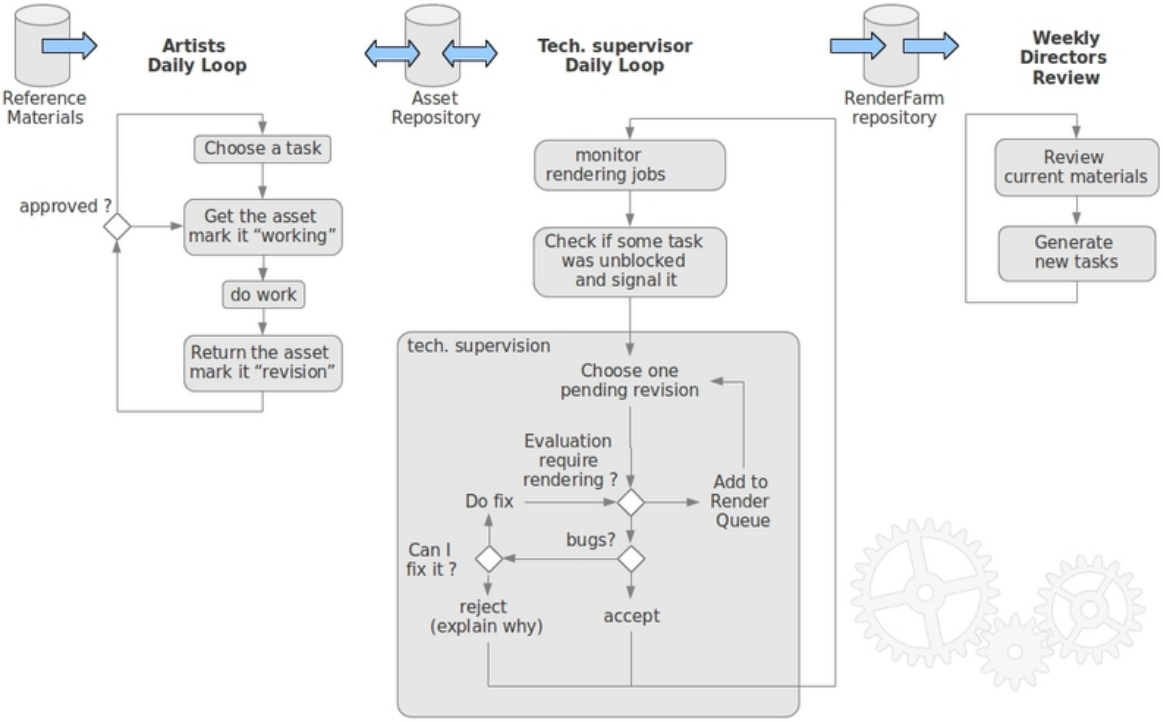


Figure 2.5 Production's workflow.

Individual Asset can be generated with different approaches:
Models:

- Reuse of existing asset: may require a format conversion;
- Acquisition from laser scan: require a post processing in MeshLab;

- Procedural Modeling: we use CityEngine, we export the generated model in OBJ, and then import them in Blender. Often these assets are geo-referenced, so we provide Blender scripts in order to import, translate and scale them in a consistent way;
- Terrain: various GIS software, in particular GRASS. The conversion to Blender follow the same rules above;
- New Models: hand crafted in Blender;
- Characters: hand crafted in Blender.

Textures:

- Use of public domain textures: mostly from http://www.cgtextures.com/
- Pictures: processed in Gimp;
- Painted from scratch: processed in Gimp;
- Commercial: sometime the use of commercial effect like animated fire and smoke can speed up the production.

2.5.3 Post Production and deployment

Each Shot is rendered on the Render Farm. All the movie frames are downloaded on a workstation for the Color-Correction and Final Editing. The speech, the music and audio effect are then inserted. This process is iterated many times during the production, and these intermediate movie are supervised by the director, the artistic director and the stereographic consultants.

Compared with the deployment of real-time applications, the deployment of a movie is an easy task. Anyway we must foresee that the movie should be prepared in different formats (low/Hi resolution, monoscopic/stereoscopic, files/DVD/BlueRay) and with different encoding so to be released trough different media (web, workstation, movie-theatre). Also it may be appropriate to provide subtitles in different language, and eventually a full speech translation.

2.5.5 Repository Metrics

We report here for completeness some quantities about the data involved in our project, in particular, our last short movie "APA: The Etruscan" that generated:

- 380 blender files, for about 6GB
- 8500 textures file, for about 7GB (some texture are animated)
- 1000 other files (animation baking, etc.) for about 1GB

- 5000 committ operation (overall)
- 3600 committ operation on Blender files
- number of committ per file: minimum=1, maximum=69 average=9.85
- frames on the renderfarm repository = 450,000 (low-res, hi-res, left, right, and their latest versions -- all JPG compressed) which sum up to 141GB

The final movie has 24,000 x 2 frames summing up to 32 GB. Finally, the coordination of the work generated more then 2,000 email, and probably more phone/skype calls. Further details on this short movie production can be found in the following paper: *A. Guidazzoli, L. Calori, F. Delli Ponti, T. Diamanti, S. Imboden, A. Mauri, A. Negri, G. Boetto Cohen, S. Pescarin, M. C. Liguori, "Apa the Etruscan and 2700 years of 3D Bologna history", SIGGRAPH Asia 2011 Posters Hong Kong, China, 2011.*

2.6 A Web application for comparative study of figurative capitals at CNR-ISTI

The Visual Computing Laboratory of CNR-ISTI has a consolidated experience in 3D acquisition of Cultural Heritage objects, from small objects to statue, from buildings to archeological sites. The 3D acquisition is performed through several techniques such as laser scanning devices (both triangulation and time-of-flight scanners), image-based techniques coming from recent Computer Vision algorithms, and other alternative acquisition form such as the Reflection Transformation Images (RTI) for the high quality visual documentation of almost planar objects made by materials that are difficult to acquire with laser devices.

The experience maturated in the use and the development of these technologies cannot be expressed by the definition of a single, generic workflow, since each acquisition methodology requires an appropriate and specific workflow. Moreover, usually the acquisition is only a step in the creation of Virtual Museum or an interactive presentation.

We decided to present here the workflow related to a research and communication project in the field of Cultural Heritage where 3D scanning plays a very important role. The name of the project is ***CENOBIUM***, whose aim was to produce an online Virtual Museum focusing on mediaeval capitals, which support functionalities for visual comparison to allow art historians and other Cultural Heritage experts to study those artifacts by means of the high detailed digital representations.

2.6.1 Concept Creation

The ***CENOBIUM*** acronym stands for *Cultural Electronic Network Online: Binding up Interoperable Usable Multimedia* (http://cenobium.isti.cnr.it). This project regards the Romanesque cloister capitals of the Mediterranean region and was designed to illustrate the cultural exchange in the twelfth and thirteenth centuries through the example of architectural decoration. To achieve this goal, the capitals of three Cloisters have been acquired so far: the cloister of Monreale (Sicily), the cloister of Sant'Orso in Aosta and the cloister of Cefalù (Sicily); extensions to other cloisters are on-going and planned in the near future. Each capital of these cloisters has been documented with a complete set of high resolution digital images and highly detailed 3D models.

This big amount of data has been integrated in a Web application where textual descriptions, high resolution images and 3D models can be accessed online in an integrated manner for each capital. Moreover, a simple and innovative visualization tool (called *LightTable*) permits to compare images and models of capitals of any cloister. The user can select them at any time during the exploration, and the tool downloads and visualizes them on the computer of the remote user on demand. In this way, it is possible to compare the selected elements, so that further similarities between the different capitals and images can be found and easily investigated.

The CENOBIUM website is a successful integration example of different types of data, in the context of an environment which is devoted not only to the wide public, but mostly to the community of experts and scholars of the Romanesque art.

2.6.2 Digital Assets Creation

Basically, two type of digital asset have been created: very high resolution images and 3D models of the capitals.

Very high resolution digital imaging

A Sinar P3 camera was used by the Photo Library of the Kunsthistorisches Institut, providing for the integration of the digital multi-shot-mode Sinarback 54 H with a resolution of 22 million pixels (sensor resolution 5440×4080 pixels), as well as various Sinaron lenses. This is a very expensive but also very high quality image capturing device, which can produce impressive results if used by a professional photographer.

The high-resolution digital images are created in a two-step process. First, a digital image is produced with a color management tool by Gretagmacbeth, following the intent to save as much information as possible with the one-, four- or sixteen-shot (taken at different exposure levels). This master copy is used for producing

further copies and for long-term preservation. Its size ranges from approximately 130 up to 520 Megabytes (TIFF format uncompressed, 16-bit colour depth and 300dpi).
A working image copy is then created from this master. This copy is digitally enhanced to allow improved quality on a low-dynamic range output device (screen or printing device). Its size is approximately 65 Megabytes (TIFF format uncompressed, 8-bit-per-channel colour depth, 4000 × 4000 pixels - approximately 33 cm on a 300 dpi printout).
The photographic campaigns on the cloisters were performed by acquiring 8 images for the more valuable capitals and just 4 images for the low interest capitals. This brought to the acquisition of a total of 690 images for the Monreale cloister, 270 images for the Aosta cloister and 180 images for the Cefalù cloister.

3D acquisition of the capitals

The more valuable capitals of two cloisters (Monreale and Cefalù) were digitized in 3D with a Konica Minolta VI 910 laser scanner (a device based on optical triangulation).
Triangulation scanning permits to acquire accurately geometry of an object with a sampling density of around 10 samples/sq.mm. and a sampling error lower than 0.05 mm. Since the scanner works at a distance between 50 and 100 cm from the objects, it was necessary to put the scanning unit on scaffolding.
It is well known that scanning any 3D object requires the acquisition of many shots of the artefact, taken from different viewpoints, to gather geometry information on all of its shape. Therefore, to perform a complete acquisition several range maps were needed; the number of range maps requested depends on the surface extent of the object and on its shape complexity.
A number from 120 to 200 single scans (each scan samples around 0.3 Million points of the surface) was needed to cover the entire surface of each Monreale capital. In the case of Cefalù capitals, due to their smaller size and more simple geometry, a number from 50 to 80 was necessary. The Monreale scanning campaign (February 2006) brought to the acquisition of 20 out of the more than 100 capitals of the cloister (Baracchini et al, 2006), with a total of nearly 4000 range maps. The Cefalù scanning campaign produced ten 3D models, with a total of nearly 800 range maps.
The acquisition of the geometry of the Aosta cloister capitals was performed using a Menci ZScan, which employs pairs of images taken from calibrated positions of the digital camera to produce accurate range maps. This technology heavily relies on Computer Vision techniques for 3D reconstruction from images. The main difference with the Minolta device is the final accuracy of the geometry that is slightly lower. Moreover, the numbers of samples are hundreds of thousands instead of millions. Using this technology, eighteen 3D models were produced for the Aosta cloister, starting from 24 range maps for each capital.

The range maps from both Minolta and Menci ZScan devices were processed with MeshLab to produce a single, complete and non-redundant 3D model. The processing steps followed the usual 3D scanning pipeline [Bernardini2002]:

- range maps alignment;
- range maps merging (or fusion), to build a single, non redundant mesh out of the
- many, partially overlapping range maps;
- mesh editing, to improve (if possible) the quality of the reconstructed mesh;
- mesh simplification and conversion into a multiresolution representation;
- color mapping (see in the next for further details).

These processing steps were executed by means of MeshLab and other proprietary software by CNR- ISTI, which give the possibility to deal with a large number of range maps and to produce the final model with the lowest possible human intervention.
The number of triangles of each model ranges from 4 to 6 millions, depending on the shape complexity and size of each capital.

Adding color to the 3D geometry

Color mapping is an important step in the scanning pipeline. As a result of the acquisition campaigns we had high quality 2D and 3D data: the objective was to integrate them in a unique model, preserving the detail of both color and geometry. In order to produce a detailed colored model starting from the set of photos provided, two phases are necessary:

1) each photo has to be "aligned" to the model: the extrinsic (position in the space) and intrinsic (focal length and lens distortion) parameters of the camera which took the photo were estimated with a semi-automatic tool [Franken2005];

2) due to the highly detailed geometry, we chose to represent color following a per-vertex approach: for each vertex, the color assigned is computed as a weighted sum of the contributions of every photo which framed that vertex [Callieri2008].

Following this approach, we produced a set of high detailed colored models of the capitals.

Tools used:

- *Color fidelity improvement of the acquired images* : Photoshop;
- *Geometry and color acquisition:* Many of the functionalities of the proprietary tools mentioned for the creation of colored geometry are nowadays integrated in the *Meshlab* open source software tool.

Additional features about geometry-image alignment for color mapping will be soon available in the next releases (planned for distribution on March 2012).

2.6.3 Concatenation of data into story

The aim of the CENOBIUM online Virtual Museum is to support scholars' study of capitals through visualization and comparison; thereofore, the data available are not presented through a story but by means of an integrated web-based system, as described in the next paragraph.

2.6.4 Visual Presentation and Interaction

The presentation of each Cloister is done through a website. For each Cloister, a brief description is provided. The most interesting section of the website is the one presented in the *Capitals* page (see Figure 2.6).

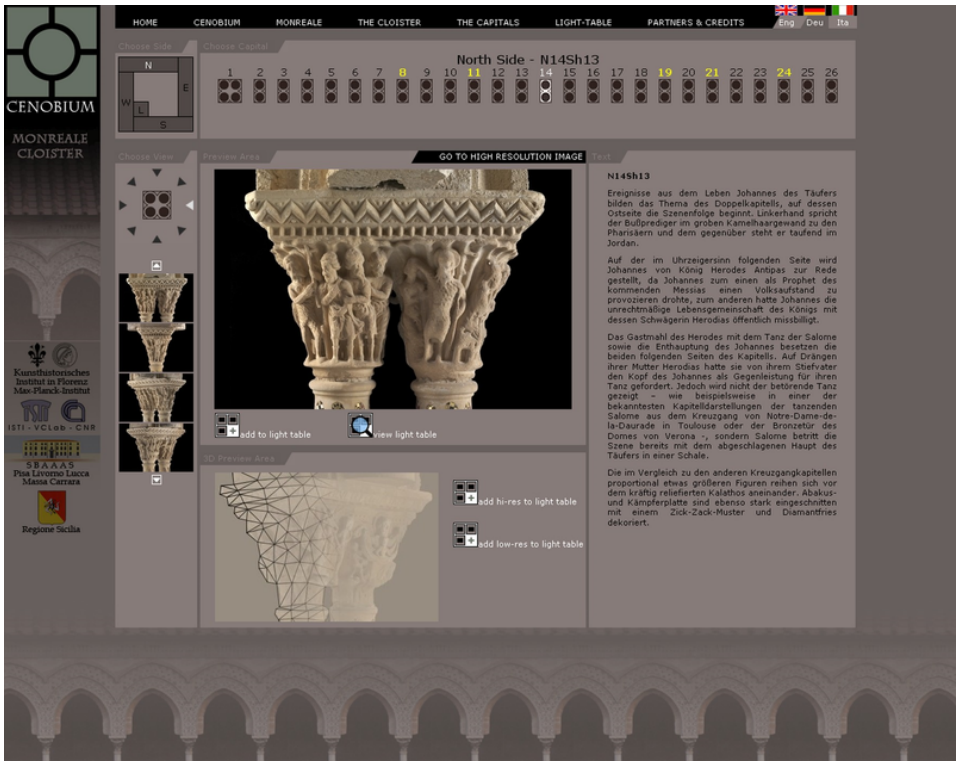


Figure 2.6 The *capitals* page where the information and the multimedia items of a cloister are shown in an integrated manner.

In this section, the users can explore in an integrated manner all the multimedia information related to each capital. After a capital of the Cloister is chosen, all the information is shown in a single web page. In order to be able to choose the capital of interest, in the upper part of the page a stylized version of the Cloister is shown; the capitals are arranged as in the real cloister, and their position is highlighted. The left part permits to navigate through all the images coming out from the photographic campaign: in particular the user can choose the side of the capital to visualize. By clicking on the selected view of the capital, the corresponding full resolution image can be visualized. Since the resolution of such images is very high, a dedicated image server gives the possibility to navigate them interactively (using a multi-resolution image representation). A textual description containing historical information is given in the right part. If available, the description can be expanded to shown a more detailed one. The availability of a 3D model representing the chosen capital is shown in the bottom part.

Both the 3D model and the images can be selected as items that can be downloaded and feeded into the LightTable tool (see Figure 2.7), by clicking the appropriate option. The ***LightTable*** tool has been recently re-designed, to endorse WebGL and to allow to manage the visualization of images and 3D models directly inside a standard web page (therefore, the download and installation of a specific browser is no more needed, as it was in the first version of the Cenobium system). When an item is selected, the LightTable takes care to download the item (high-resolution photos or 3D geometry) on the local PC. Then, for whose items that are ready-to-use, the user has the possibility to visualize them simultaneously for comparison. In this way, further similarities between the items can be found and investigated easily. The maximum number of items that can be visually compared is four.

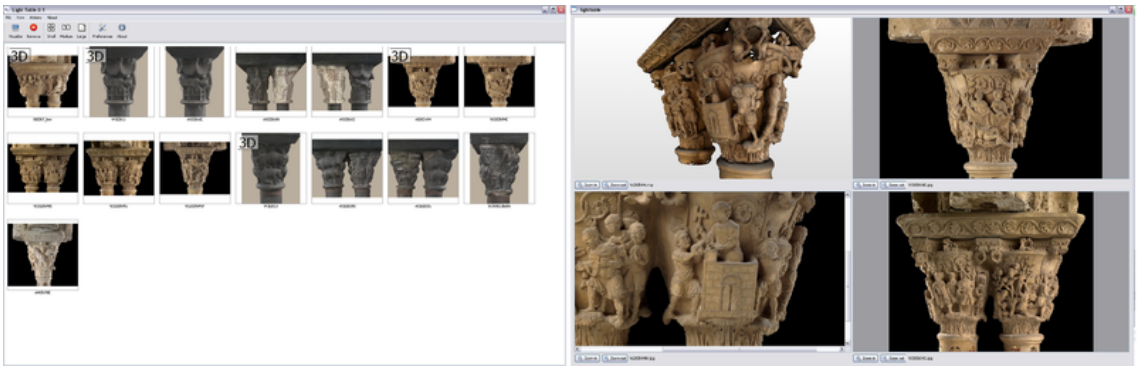


Figure 2.7 The LightTable in action. (Left) A selection of images and 3D models. (Right) A visual comparison session.

2.7 Noho workflow

The ***Medieval Dublin*** project (see Figure 2.8). is the biggest Noho undertaking to date, so we introduce this production as our main example. This was also our first project in the production of virtual presentations for the CH domain, so naturally we encountered a few cul-de-sacs and bad practices along the way. Consequently, some more recent projects (notably The Abbey Theater project and iDiscover Lorrha) are cited as well to propose some examples of where we have improved that pipeline.

Figure 2.8 A screenshot from a video of the Medieval Dublin project (web site: http://www.medievaldublin.ie).

2.7.1 Concept Creation

Medieval Dublin was commissioned to Noho by the heritage officer of the Dublin City Council. They enlisted a scientific panel or what we called a steering committee of notables in the field of History, Archaeology, paired by representatives of museums in Dublin. Several meetings took place in the development of the project (there were two phases, Edition 1 and Edition 2), aimed at presenting basic material and intermediate results for historical and communication validation. The advice and information gleamed from these meetings were very important to the success of the project. The initial first meetings defined the scope of the project.

2.7.2 Digital Assets Creation

Content Outline

Defining the content outline was a collaborative task carried out mainly by Noho staff and involving both the narrative point of view and the visual perspective. Once the wishes of the Steering Committee were made clear, we began to rough out the content outline consulting with our internal archaeologist. Her involvement was

crucial in that she could outline what stories were interesting and important and what direction the reconstructions should go visually.

3D Modelling and Animation

The 3D work on this project has been completed over a very long period of time and was improved and re-structured in the second edition. All our 3D models in the second edition were produced with Softimage XSI. There is no laser scanning involved in the project. The 3D models can be divided into the following categories:

- Timeline Model
- Specific Reconstruction Models
- 3D Special Effects (Fx) Scenes
- Character Animation Scenes

The Timeline Model was a 3D scene that animates the evolution of the city of Dublin from year 800 AD to 1540 AD. The scenes timeline in frames is set accordingly, so when a tower for a cathedral is built in one year it pops on that frame. This makes it easy for anyone working on the scene to see what is happening on each date.

Two main sets of house types were defined:

- the viking period which had 8 different houses (taken from Pat Wallace's excavation of Fishamble St.);
- the later medieval house types, where we looked to England for examples (only one survived in Dublin).

There were then walls, defensive structures and ecclesiastical buildings to create. These were done individually and then brought into the scene. The landscape and animated river and terrain was done using various tricks in 3D.

Models were imported and exported throughout the different scenes and versions and naming was a little haphazard. To improve this in future we would use referenced models, however these have limitations if you are editing the models specifically in that scene.

Similarly, there was no meta-data associated with any model. In one of our Timeline scenes, we began to use an annotation function for some of the models and this is something we will use in the future. More sophisticated handling of meta-data is for us a major need.

2.7.3 Visual Presentation and Interaction

Video Composition and Editing

All Video Direction and Editing was handled by the team. There was a lot of grading and compositing so that the shots from various different animators were kept consistent. In the editing process storyboard frames mixed with wireframe moves were used initially and then replaced as the shots progressed. The naming process for shots and passes of shots is always a tricky one and we are still looking for a improved solution.

Design and deployment

The logo and packaging of the product itself was designed in house. This was then carried through into an iPhone App and any promotional material. The brand design covered video, web, print and interactive content. Graphic design and illustration were also needed for various parts of the animation and interactive elements. There was an interactive DVDrom, a website, an iPhone app and a related custom-designed kiosk also.

Interactive Development

Once the material is nearing the final versions, we have began to think to the possible production of an interactive application from the material developed for the passive video application produced.

2.8 Etruscanning

2.8.1 Concept Creation

The goal of the Etruscanning project is to show two Etruscan tombs in their original state, at the moment that the tombs were closed (see figure 2.9 for a photo of a virtual installation). This means that both the tomb and the objects in the tomb need to be shown as they were without any damage or corrosion.
This project is developed by a subset of the V-MUST partners and consists of the following parts:

- virtual reconstruction of the original Etruscan tombs
- digitisation of the physical tombs and of the objects retrieved (now conserved in a museum)
- digital restoration of the museum objects and of the tomb architecture
- creation, translation and recording of storytelling
- creation of the digital application that uses all these assets
- installation in different museums of the communication installations

- evaluation of the museum installations and feedback to developers
- update and maintenance of the installations

Some of these parts are executed by one of the partners, in collaboration with external experts, at four different locations in Europe. Most parts are even divided over different partners at different locations. Currently, the exchange of data happens in a variety of ways: by ftp-server or by sending hard disks, DVDs, emails. The documentation of the process is scattered over a blog, emails, Word files and Excel files; unfortunately, some part of the process are not documented at all. A lot of knowledge about the workflow and the methodology is implicit and only in the heads of the participants.

Figure 2.9. Etruscanning 3D project. The virtual reality installation dedicated to the digital reconstruction of the Regolini Galassi etruscan tomb in Cerveteri.

2.8.2 Digital Assets Creation

For the Regolini-Galassi tomb, the objects were extensively photographed in the Museo Gregoriano Etrusco in the Vatican, to produce from this photographic campaign:

- basic 3D models (through the use of image-based reconstruction techniques);
- Object VR (for use in multimedia systems);

- and finally standard documentation (to derive the details of the objects)..

These photographs were delivered by the photographic department of the Vatican as high resolution TIFF images on DVD. Visual Dimension and CNR-ITABC used the images to make 3D models (dense stereo matching) and unwrapped textures (different 2D and 3D techniques were used). These unwrapped textures were used by Visual Dimension to make digitally restored depth maps, that were applied as normal maps by CNR-ITABC within Unity 3D.
To verify the 3D result of this digital restoration process by Visual Dimension and external experts, a Unity 3D visualisation module was developed. Once approved, the digitally restored objects are positioned correctly in the tomb, based upon a dummy 3D model that was the result of an extensive virtual reconstruction process.
Conversion of formats, to link the different parts of the reconstruction process to each other, are crucial (but mostly done by tools within the software platforms that created the data, not by external tools). The 3D data of the virtual reconstruction process (which used dummy objects) had to be converted from ArchiCAD (.pln) to 3D Studio Max (.3ds) for integration into the Unity 3D platform. The photographic documentation, provided by the Vatican Museums, was provided in TIFF format (hence taking about 80 % of the total data volume) and had to be converted to JPEG to feed the 3D dense stereo matching processes to create 3D models (in .ply) or to create Object VR movies for the multimedia systems that will support the 3D VR application. The digital restoration created .max files (through manual 3D modelling) or used depth maps (created as .psd files in Photoshop, exchanged as b/w JPEG uncompressed) that were converted to normal maps within Unity3D.
The exchange of the data between the different groups and project parts is an unstructured ad-hoc process, mainly based upon email exchanges. This means that no other team could take over the current project, not now or in the future. In this project, none of the conversion was done through external tools (except some TIFF to JPEG bulk conversion), but through the integrated functionalities of the creation software used.

2.8.3 Concatenation of data into story

A large body of research results in the form of scientific publications, images, historical documents or specific research was produced to underpin the virtual reconstructions and storytelling, amended by two external domain experts where needed.
A part of this process was documented through a public blog (find at the following link http://regolinigalassi.wordpress.com) while another part of the decision process resides in emails. A part of the process (for example most of the creation of the storytelling text) is not documented explicitly. It needs to be noted that a public blog does not work for scholars, since no comments where produced on the blog but remarks were send by email. This probably has to do with the fact that scholars consider a blog as a publication, and not as an exchange of opinions, hence as research.

Most of the production chain is not documented at all, but done through ad hoc processes of 3D creation (manual or through dense stereo matching), digital restoration (through Photoshop or 3D editing), processing of laser scan data (cleaning, merging, texture mapping) or creation of the interactive VR application (animation, combination with storytelling).
A small part of the methodology was documented through Word-files or entries in the project blog. Most of the methodology is simply not documented at all. This has several implications. First of all, to be credible within the museum domain, we need to be able to proof the scientific accuracy of the digital data (for example as result of digital restoration). This means that we need to have full traceability of creation, editing and decision processes at all times. Additionally, many digital assets of the Etruscanning application went already through an updating cycle as the partners continue to improve the content and the functionality of the existing VR application. We need to be aware that digital assets do not have the invariability of most museum objects. Digital museum assets have complex creation processes and can continue to change through updating and adaption to different uses and contexts.

2.8.4 Visual Presentation and Interaction

Once positioned, the objects are animated based upon interactive input from the Kinect device. Etruscanning is not a single installation project. During the two years of the project, the VR application is permanently improved while the digital assets are updated through innovative techniques of digital restoration. The VR application will be installed in 6 museums, each time within a different context, and complemented by other multimedia systems that use other instances of the digital assets. This means that VR application and digital assets are re-used several times, hence the need for a proper documentation.
The Etruscanning application will be used from 2013 onwards by three museums on a permanent basis (Museo Gregoriano Etrusco in the Vatican; Villa Giulia Museum, Rome; and the Etruscan Museum in Formello, Italy). This has two major consequences. First of all, it is very unlikely that all three installations in those three museums will be identical, so each installation will be a different implementation based upon the same data. Secondly, each of these permanent installations will need to be maintained for a long period, as they will be an important part of the permanent exhibitions of these museums.

3 User Interface Design

This chapter deals with best practice for user interface design. The chapter starts with a an introduction to common usability definitions and examples of usability goals. Then follows an overview of general principles for user interface design that can be considered fundamental for all sorts of Virtual Museum user interfaces. The chapter is concluded with more specific design principles and guidelines for 3D user interfaces.

3.1 Usability

Usability can be regarded as a way to assess the quality of the interaction between a product and the user. Unlike other terms, such as "user friendliness", it has been considered important with a clearly stated definition, in order to conduct evaluations and compare results to specified usability goals. One of the earliest definitions of the term was stated by Brian Shackel (1986), and was based on four key-headlines:

- **Learnability**: the relation of performance to training and frequency of use, i.e. the novice user's learning time, and relearning on the part of casual users.
- **Efficiency**: the results of interaction in terms of speed and errors.
- **Attitude**: acceptable levels of tiredness, discomfort, frustration and personal effort.
- **Flexibility**: allowing adaptation to tasks and environments beyond those first specified.

Another widely acknowledged definition is stated in ISO 9241-11: The effectiveness, efficiency, and satisfaction with which specified users can achieve specified goals in particular environments.

- **Effectiveness**: the accuracy and completeness with which users achieve specific goals.
- **Efficiency**: the accuracy and completeness of goals in relation to resources expended.
- **Satisfaction**: the comfort and acceptability of the system.

The above stated key-headlines can serve as a starting point when specifying relevant *usability goals* for a specific product. For instance, regarding "Efficiency", one such goal for a certain product could be formulated as: 80% of the users must be able to finish the installation within 5 minutes. Another example regarding "Attitude" could be that the average rating for perceived comfort should be between *Good* and *Very good*.

3.2 General principles for User Interface Design

There is a vast amount of guidelines for user interface design on different technology platforms, e.g. *iOS Human Interface Guidelines* and *Windows User Experience Interaction Guidelines*. Section 3.4.2 will therefore focus on general *principles* for user interface design. While guidelines are narrowly focused (e.g. on a particular platform), principles are more fundamental, widely applicable and enduring (Shneiderman & Plaisant, 2010).

(a)

(b)

(c)

(d)

Figure 3.1 (a) Affordance: Conflict between actual and perceived affordance. The door is opened by *pushing* but its handle signals that it should be opened by *pulling*. (b) Visibility: less used or more advanced functions are often hidden in the menu systems of digital cameras. (c) Mapping: the control knobs of an adjustable car seat are shaped to resemble the part of the seat that they control. (d) Feedback: when removing an e-mail, the iPhone user gets very clear feedback on the result of this action.

3.2.1 Affordance, visibility, mapping and feedback

In his book *The Design of Everyday Things* (Norman, 1986) Donald Norman presents a set of design principles that he believes must be considered in order to achieve a product with good usability. Fours of the design principles are presented below:

- **Affordance** is the actual and perceived action possibilities of an object perceivable by a user. In the design community the term affordance is mainly used to describe the *perceived* action possibilities. *Example A GUI button with a three-dimensional look affords pushing.*
- **Visibility** is about making relevant parts of the user interface visible. *Example Settings for rendering style, resolution etc. is put in a slightly hidden preferences menu.*
- **Mapping** is the relationship between a certain property and the function of an object. *Example The "Start Virtual Museum" button is made big and green so that the user immediately understand its importance.*
- **Feedback** is about giving each action in a user interface an immediate and obvious feedback. *Example When the user pushes the "Load 3D model" button on the screen, a progress bar immediately appears on the screen, telling him/her how much of the model has been loaded.*

3.2.2 Conceptual models

Even though today's technology offer more and more interaction possibilities (e.g. gesture-based interaction), *a good conceptual model* remains one of the most important aspects in user interface design (Norman, 2010). A conceptual model is a framework, a context, and reasons for understanding (Norman, 1986). One of the most famous conceptual models is the notion that working with a computer is like working in an office, a model that basically all modern operating systems are built on. Another example is the underlying conceptual model for most web shops: shopping online is like buying things in a real shop with actions like "add to basket" and "checking out". *Example Malmö city archive wants to make their vast 17th century collection accessible*

through a online Virtual Museum. The designers decide to use the conceptual model of visiting a library with actions like "keyword search" and "read/examine books".

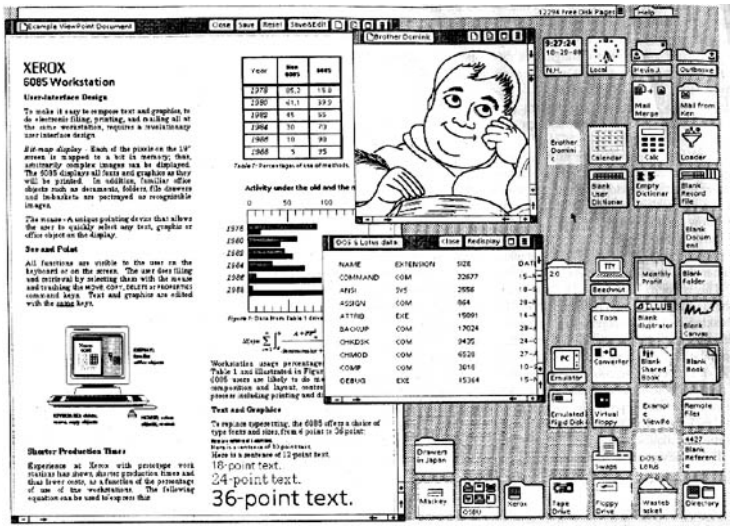


Figure 3.2 A famous conceptual model: The Xerox Star user interface (1981) introduced the idea that handling a computer is like working in an office.

3.2.3 User Interface Metaphors

One way to build a good conceptual model is to choose proper user interface metaphors. For example, the desktop metaphor is used to communicate the conceptual model of working in an office. The idea behind metaphors is to connect the source domain (e.g. office work) with the target domain (e.g. computer work). They must be chosen and designed with great care since a poor metaphor can compromise a tool's usability. One fundamental problem is that they can communicate both *too much* and *too little* to the user. For example, using a floppy disk icon for a saving function might confuse or annoy younger users (Figure 3.3a). Similarly, a book shelf metaphor might make the user less prone to look for a search function in the user interface (Figure 3.3b). *Example Part of the conceptual model of* Malmö Archive online Virtual Museum *is a search function based on a librarian metaphor and a reading room metaphor with tools for facilitating reading, such as a magnifying function and various lighting settings.*

(a)	(b)

Figure 3.3 Two common user interface metaphors: (a) The disk metaphor. (b) The bookshelf metaphor.

3.2.4 Jacob Nielsen's Ten Usability Heuristics

The ten usability heuristics are ten general principles for user interface design developed by Nielsen (1994). They are useful not only for designing user interfaces but also for making so-called *heuristic evaluations* where usability specialists judge whether each element of a user interface follows a list of usability heuristics.

- **Visibility of system status**
  The system should always keep users informed about what is going on, through appropriate feedback within reasonable time. *Example When John pushes the "Load 3D model" button on the screen, a progress bar immediately appears on the screen, telling him how much of the model has been loaded.*
- **Match between system and the real world**
  The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order. *Example Martin pushes the Library icon to access the written source material on which the* Malmö 1692 Virtual Museum *is based.*
- **User control and freedom**
  Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Example By mistake Lisa pushes the exit button in the Medieval Dublin Virtual Museum*. Luckily, a confirmation dialog box allows her to go back to the application.*

- **Consistency and standards**
  Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions. *Example John recognizes the file explorer inside the* Iron-age Lund Virtual Museum *from Windows and immediately understands what he can use it for.*
- **Error prevention**
  Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action. *Example Martin finds it very easy to push the right button on the touch screen of the* Flaminia Reloaded Virtual Museum *since the buttons are big and clearly separated.*
- **Recognition rather than recall**
  Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate. *Example Accessing the main menu of the* Mercati di Traiano Virtual Museum*, Marco sees ten different stories that he can choose among.*
- **Flexibility and efficiency of use**
  Accelerators - unseen by the novice user - may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions. *Example Lisa quickly jumps between the fields of the the quiz form of the* Medieval Rome Online Virtual Museum *by using the Tab button.*
- **Aesthetic and minimalist design**
  Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility. *Example Matt has no problems understanding the text instructions on the Virtual Museum screen since it is clearly presented.*
- **Help users recognize, diagnose, and recover from errors**
  Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution. *Example Isabella immediately understands that she has entered the wrong user name for logging in to the* Virtual Troy online Virtual Museum *when the error message appears.*
- **Help and documentation**

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large. *Example John is a bit unsure on how to change the navigation from flying to walking in the* Sarajevo Virtual Museum*. A context sensitive help function gives him a tip for how to proceed.*

3.3 3D User Interfaces

A 3D user interface *(*3D UI) is a user interface that involves 3D interaction, i.e. human-computer interaction in which the user's tasks are performed directly in a 3D spatial context (Bowman, Kruijff, LaViola & Poupyrev, 2004). A 3D UI offers interaction by means of *3D interaction techniques*, i.e. combinations of hardware and software elements that provides a way for the user to accomplish a single task in such a 3D context (Figure 3.4). Bowman et al (2001) has suggested a taxonomy for 3D interaction techniques based on what tasks they support:

- **Navigation**
  - Travel (the motoric component of navigation)
    Example A user is handling a joystick to move closer to the Parthenon temple.
  - Wayfinding (the cognitive component of navigation)
    Example A user is trying to find his way inside the Pompeii Virtual Museum.
- **Selection & Manipulation**
  Example A user selects a clay figurine in the Cyprus Virtual Museum and examines it by rotating it.
- **System control**
  Example A user is changing the rendering mode to sketchy in the Via Flaminia Virtual Museum*.*

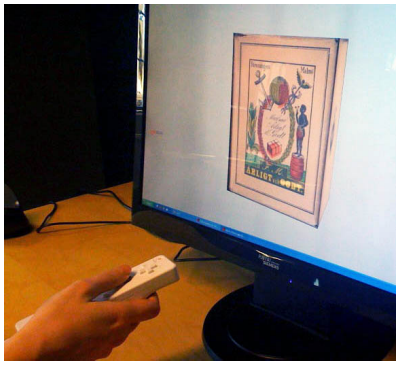


Figure 3.4 An example of a 3D user interface for a Virtual Museum: the user can select and then manipulate 3D artifacts with a Nintendo *WiiMote*.

3.3.1 Navigation

3.3.1.1 Immersive Systems

Bowman, Kruijff, LaViola & Poupyrev (2004) have formulated a set of guidelines for navigation in virtual environments. The guidelines tend to focus on more immersive VR systems capable of tracking the user's movements. Nevertheless, they are to some extent also applicable for other sorts of 3D applications, such as videogames. The navigation guidelines with most relevance for 3D UI:s inside virtual museums are presented below.

3.3.1.1.1 Travel guidelines

- **Match the travel technique to the application.** *Example Matthew gets a quick overview of the Parthenon temple by circling around it with the orbital viewing mode.*
- **Consider both natural and magic techniques.** *Example Maria can quickly move between different places in the* Virtual Museum of Iraq *by travelling with a flying carpet.*
- **Use an appropriate combination of travel technique, display devices, and input devices.** *Example A pen-shaped interaction device gives John the precision he needs to place the viewpoint exactly where he wants it inside the* Iron-age Lund Virtual Museum.
- **Choose travel techniques that can be easily integrated with other interaction techniques in the application.** *Example By interacting with a virtual 2D map inside the Pompeii Virtual Museum , Maria can both "teleport" to other locations in 3D Pompeii and she can also select and manipulate 3D artifacts.*
- **Provide multiple travel techniques to support different travel tasks in the same application.** *Example Martin can explore the House of the gladiators inside the Pompeii Virtual Museum thanks to the Kinect head tracking. He then moves on to another house by pointing on a virtual 2D map.*
- **Make simple travel tasks easier by using target-based techniques for goal-oriented travel and steering techniques for exploration and search.** *Example Lisa can travel between different key locations in the Medieval Rome Virtual Museum by interacting with a virtual 2D map (Figure 3.5).* The Flaminia Reloaded Virtual Museum, *instead, offers her free exploration by means of a joystick.*
- **Use graceful transitional motions if overall environment context is important.** *Example When Martin points at the Temple of Jupiter on the virtual 2D map, the Pompeii Virtual Museum*

moves him there with a quick but smooth movement, giving Martin a good understanding for the spatial layout of Pompeii.

- **Consider integrated (cross-task) interaction techniques if travel is used in the context of manipulation.** *Example Isabella can select and manipulate artifacts with gestures in an 3D excavation inside the Virtual Troy Virtual Museum . In the same way, she can also move the viewpoint by manipulating the excavation object itself.*
- **Desktop 3D navigation techniques should allow the user to accomplish the most common travel tasks with a minimum of effort.** *Example The main purpose of the desktop version of the Iron-age Virtual Museum is to let the user quickly get a feeling for the spatial layout of the Iron-age city Uppåkra by controlling a joystick.*

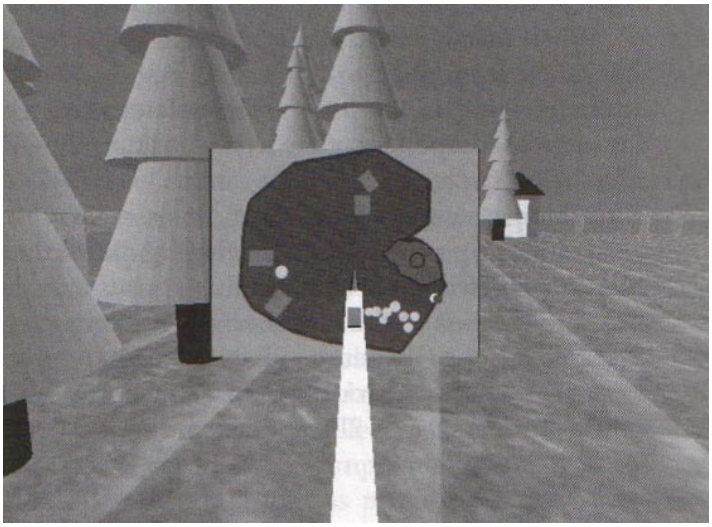


Figure 3.5 Target-based travel using a 2D map: the user can drag a dot representing his/her current position to a new location on the map to define a travel target in the virtual environment (Bowman, Johnson & Hodges, 1999).

3.3.1.1.2 Wayfinding guidelines

- **Match the motion cue to the task.** *Example A virtual 2D map visualized as a piece of parchment is used as a wayfinding aid in the Medieval Rome Virtual Museum.*

- **Match the motion cue to users' skill.** *Example John has very little experience of virtual environments and therefore finds the accentuated landmarks very useful for finding his way through the virtual version of 17th century Malmö.*

- **Don't make motion cues dominant features.** *Example The Eternal Egypt Virtual Museum has been equipped with very subtle wayfinding cues, e.g. sign posts, to avoid that its users only move between landmarks.*

- **Choose input devices providing real motion cues if possible.** *Example A treadmill is used as input device in the Medieval Dublin Virtual Museum to facilitate the wayfinding for the users as much as possible.*

- **Avoid teleportation.** *Example When Martin points at the Temple of Jupiter on the virtual 2D map, the Pompeii Virtual Museum moves him there with a quick but smooth movement.*

- **Integrate travel and wayfinding components.** *Example A small 3D map in the field of view makes it possible for Anna to select targets to travel to inside the Troy 3D Virtual Museum, while at the same time providing a good overview of the virtual environment.*

3.3.1.2 Non-immersive Systems

When it comes to navigation in non-immersive systems (e.g. desktop applications), it is informative to look at the video-game industry and the virtual world community. Navigation is usually achieved through the combination of pointing device and keyboard. The A, S, W, and D keys are almost universally being used for left, reverse, forward and right movement. Additional redundancy is often built in though the use of the cursor arrow keys to duplicate these motion functions. While the forward and reverse triggers usually move the user from one position to another, the function of left and right triggers varies between platforms; sometimes also causing change of position (strafing), but sometimes simply causing the avatar to rotate on the spot (i.e. to yaw on the vertical axis). Conventions for vertical motion are even less stable, although the Page Up/Down keys are often used for the functions of vertical assent and descent respectively; together with the keys for forward and reverse, left and right, these environments provide the user with a full set of six degrees of freedom in movement. Other keyboard-triggered functionality such as crouching or jumping, which are common in video games, seldom have similar equivalents in cultural heritage implementations of virtual world platforms (even if the ability is potentially technically present within the platform). When designing the keyboard interaction, the designer must be aware of both the ergonomic arrangement of keys and any additional complications that may

arise from the use of compact or regional or language-specific keyboards, as well as differences between keyboards produced by different manufacturers, most notably PC and Apple Macintosh.

A pointing device is often used in combination with the keyboard to control the camera view, that is to say the "eyes" of the avatar. As with keyboards, the variety of pointing devices that is commonly available, as well as the range of device-specific functionality, must be kept in mind. It cannot be assumed that the user will have, for example, a two-button mouse, and so it may be unsafe to rely on users having access to a right-mouse button. Like keyboards, pointing devices can be customised by the user and this should be considered when designing interaction through such devices.

Within first-person implementations, the camera controlled by the pointing device (translating the device movement into horizontal and vertical rotation, and pitch and yaw of the camera) usually imitates the physical constraints of human vision by constraining the movement of the camera to the determined eye-level of the avatar, and to ca. 90° horizontal and ca. 60° vertical from a centred, frontal gaze. Such constraints are especially desirable where the presence requirement is high (that is where the implementation is aiming to achieve an immersive experience or is attempting to mimic the physical restraints of human neck movement) and especially where the user's avatar is graphically represented within the world, preventing the user from seeing their own body parts. Full 360° head movement in such cases is achieved though the combined use of keyboard and pointing device.

While a first-person perspective may be seen as being the best option for aiding a sense of presence, the use of the pointing device for both controlling the user's view and as a means for accessing other fixed-position, on-screen components such as menus may lead to confusion as the viewpoint moves to follow the cursor's movement towards on-screen components. This is especially problematic if the interface is designed to enable users to perform functions by interacting with objects within the world. Design should incorporate modal operation - i.e. the system should temporarily switch from one mode to another, e.g. from using the pointer to set camera direction to using the pointer to select interface elements. Users may access a different mode through, for example, additional input via the keyboard, or by building into the user interface design a degree of purposeful redundancy.

Where the user's view onto the virtual world is not first person, the camera typically floats behind and slightly above the avatar, enabling the user to see both the virtual world and their own presence, as avatar, within that world, as they move through it. In some platforms, users may be able to alter the distance at which their

camera follows their avatar, but some platforms even additionally allow users to move their camera's viewpoint independently of their avatar, for example zooming in on details or far-away objects without the avatar's body moving at all. This functionality is crucial, indeed, to the Second Life and Open Simulator platforms' capacity to enable users to build three-dimensional content within the world: builders need quickly and fluidly to view their content from several angles as they work (Figure 3.6). In these two platforms, free camera movement mode is activated by holding down a certain key on the keyboard; the user can then use the pointing device to select a point within the virtual world, and then use either navigation keys or the pointing device to rotate around the selected point, or to move closer or farther away from it.

Figure 3.6 A *Second Life* user is making use of the free camera mode when designing an historical reconstruction.

A free camera approach provides the user with greater flexibility in the way in which a virtual space can be explored, but at some cost, perhaps, to immersion in an avatar-centric experience of the world; the sense of virtual embodiment that aids users' imaginative entrance into the fiction of being present within a space. The disembodied camera's out-of-body experience produces a different set of user expectations; the capacity of this virtual "eye" to enter spaces and access vantage points that an avatar would struggle to access means that many of the techniques conventionally used to optimise the graphical performance of a real-time environment, such as simplifying geometry or omitting textures of features in areas that an avatar cannot see, may not apply. Other consequences of free-camera viewing include: the possibility of transecting geometry as the camera passes through "solid" objects (which, again, can be more easily anticipated and minimized in first-person systems); errors in implementing point selection (such as ray casting, which relies on the translation of the

position of the point selected on the two- dimensional screen into the three dimensional world space, calculating a vector, or ray, from the camera poison to the notional position of point selection. This process can be flawed by calculation errors and the interruption of the vector by other unexpected world objects, causing users who are attempting to select an object, to miss or over/undershoot the desired target, or accidentally to select objects, especially transparent (though "physical") objects or interface elements.

3.3.2 Selection & Manipulation

One of the fundamental ideas behind Virtual Reality is to allow the user to interact directly with the virtual environment and objects. This has also been a vision in the domain of videogames that has started to become true the last five years. For example, in a modern videogame such as *The Elder Scrolls V: Skyrim* the user can interact directly with just about anything in the 3D game world.

3.3.2.1 Immersive Systems

Bowman, Kruijff, LaViola & Poupyrev (2004) have formulated a set of guidelines for selection and manipulation navigation in virtual environments. The guidelines tend to focus on more immersive VR systems capable of tracking the user's movements. Nevertheless, they are to some extent also applicable for other sorts of 3D applications, such as video-games.

- **Use existing manipulation techniques unless a large amount of benefit might be derived from designing a new, application-specific technique.** *Example John zooms in on one of the clay figurines of* The Cyprus Museum Virtual Museum *with a pinch gesture.*

- **Use task analysis when choosing a 3D manipulation technique.** *Example The design team of the* Iron-age Lund Virtual Museum *carefully investigates what different user groups want to do with the virtual objects before designing its 3D UI.*

- **Match the interaction technique to the device.** *Example The design team of the* Iron-age Lund Virtual Museum *decides to prioritize zoom and rotation of the virtual objects with high accuracy and therefore implements these interaction techniques for a 6 DOF 3D mouse.*

- **Use techniques that can help reduce clutching.** *Example* The Cyprus Museum Virtual Museum *is designed in such a way that it's enough with a small, comfortable hand gesture to rotate the 3D artifacts.*

- **Non-isomorphic techniques are useful and intuitive.** *Example Marco's movements are non-linearly scaled so that it becomes easy for him to interact also with faraway objects in the Parthenon Temple Virtual Museum* (Figure 3.7).

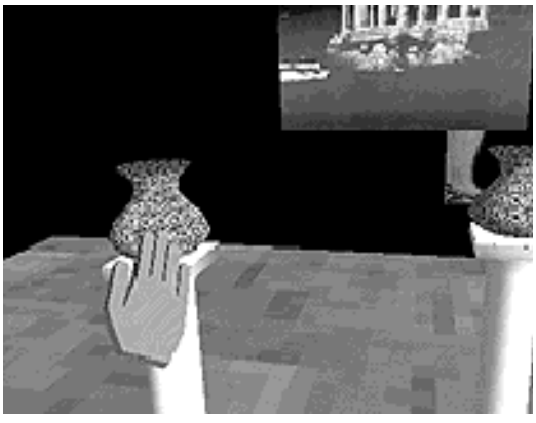
- **Use pointing techniques for selection and virtual hand techniques for manipulation.** *Example Martin selects a clay figurine in the* The Cyprus Museum Virtual Museum *by pointing at it, whereupon a virtual hand grabs the figurine. Martin can then control the virtual hand's movements by moving his own hand.*

- **Use grasp-sensitive object selection.** *Example John selects an artifact inside the* Kulturen in Lund Virtual Museum *by orientating the Wiimote so that it's orientation matches the one of the 3D object.*

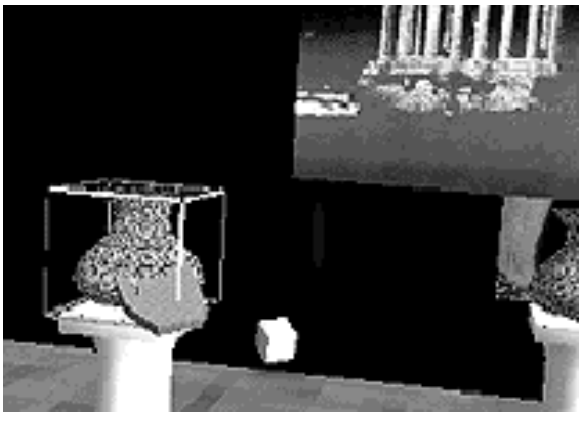
- **Reduce degrees of freedom when possible.** *Example When Maria has selected the 3D golden bracelet in the* Iron-age Lund Virtual Museum *she simply drags it to the Laboratory icon for later examination.*

- **Consider the tradeoff between technique design and environment design.** *Example The scanned excavation sites inside* The Cyprus Museum Virtual Museum *has been scaled down 20% to make it easier for the users to interact with it.*

- **There is no single best manipulation technique.** *Example The developers of the* Lund Iron-age Virtual Museum *are not completely satisfied with the way user's can manipulate the 3D artifacts. However, they know that it was the best choice for this particular application.*



(a)



(b)

Figure 3.7 The Go-Go Interaction Technique: (a) With a traditional manipulation technique the user cannot reach the vase since reach is limited by arm length. (b) The Go-Go technique allows the users to expand his reach. *Image courtesy of HIT Lab*.

3.3.2.2 Non-immersive Systems

When it comes to selection and manipulation of virtual objects in non-immersive systems (e.g. desktop applications), it is informative to look at the videogame industry and the virtual world community. Arguably, the most formidable obstacle for the developer to overcome is the connection between the user and the mechanisms used to trigger interaction with the virtual environment. This issue can be discussed in terms of "passive" and "active" interaction strategies.

3.3.2.2.1 Passive Strategies

Of the two, passive interaction, where the user does not need consciously to interact with objects within the virtual environment in order to release functionality, is possibly the simplest to implant, relying on either the location or the movement of the user's avatar or camera.

Location-based passive interaction requires the system to track the avatar's position (and possibly orientation) until certain criteria are met (such as: the avatar is within *x* meters of location *y*) at which point the trigger is activated. This method has the advantages both of being invisible to the user, and of making no further demands on the polygon count of the world, therefore not detracting from the virtual environment's graphical performance. However, location-based passive interaction has limitations. The system must constantly track the position of avatars in relation to the pre-defined trigger zones in order to monitor if a trigger condition has been met, and this incurs a cost in processing time. This cost can in some cases be moved from the server to the client, but avatar tracking rapidly becomes a large overhead if the system must simultaneously track multiple avatars.

The specified area for the trigger must also be carefully considered. The most efficient way to create a trigger zone is by defining a single point and its radius, which will result in a spherical trigger zone. If using this approach, the placement of the trigger zone must be carefully considered to ensure that it does not extend beyond the desired area, for instance, by "leaking" through a wall into neighbouring room; considerable confusion can be caused if a user in one area inadvertently triggers interactive content that affects users in an adjacent area. Whereas the architecture and layout of fictional spaces can be designed to avoid such unwanted overlaps, designers may not enjoy this luxury when the space in question represents an actual cultural heritage site, in which several points of interest may potentially arise in close proximity to each other.

Movement-based passive interaction also deploys trigger mechanisms, but it executes activation differently. Where location-based passive interaction requires the system to track and compute trigger conditions, in movement-based passive interaction, objects within the virtual world detect when an avatar approaches them. A common example is a doorway; in this method, the doorway would incorporate an

additional object designed to execute the desired function when the trigger condition – for example an avatar passing through the doorway – is met. The advantage that movement-based interaction has over location-based interaction is that no system-level, "global" tracking has to take place, because the objects themselves contain the information for the trigger zone as part of their properties.

Although methods different from platform to platform, there are two approaches to implementing movement-based passive interaction, namely: **scanning** and **collision**.

Scanning can be considered a localised form of location-based passive interaction. The object has an origin (i.e. a centre-point) and a defined zone. The object is programmed to perform a scan of the specified zone at regular intervals (the developer may be able to determine how frequently), in order to check if an avatar is within the zone's perimeter. If an avatar is detected, then the trigger condition are considered met and the interactive function may be executed. This object-based approach also allows additional criteria to be set for meeting the trigger condition.
Collision, as the name suggests, requires that an object be placed within the environment that will detect if an avatar makes contact with it, whether by walking upon, passing through or bumping into it. Collision permits the creation of much more tightly-defined trigger zones, almost eliminating the chances of "leaking". As the triggers are tied to specific points within the world, the direct consequences of interaction are more predictable for both designers and users than those of location-based methods, and this may be viewed as an advantage or disadvantage depending on the application being created.

Both scanning and collision have their drawbacks. Scanning, while less computationally intensive than location-based methods, still takes up system resources each cycle to run the script that ascertains if the perimeter has been breached, and when these scripts are used in high numbers (or when they scan – or "poll" – the zone very frequently) this tends to outweigh any system performance advantage that might be gained.
Collision techniques do not have such a high computational burden, but may require additional items to be placed into the scene for avatars to collide with, thereby increasing the polygon count or (even if rendered as "invisible" objects) contributing to the graphical processing requirements. In addition, as noted earlier, invisible objects may interrupt other interaction methods, such as ray-casting for object selection.
The functions that constitute the trigger conditions must also be carefully considered. When scanning for avatar proximity, the specified parameters will always be met as long as the avatar is present within the trigger zone; and when calculating collision, the conditions will be considered met for however long an avatar maintains contact with the collision object; either methods can therefore cause unexpected effects if an avatar lingers within, or in contact, with a trigger mechanism. While some platforms allow designers to define "on avatar

enter" or "on avatar exit" as conditions for scanning or collision triggers, these add further complexities to both design requirements (such as the need to define entry and exit positions) and the system's real-time processing requirements (such as the monitoring of trigger status in complex conditions).
To apply a movement-based method to a doorway, for example: the condition is met when the avatar passes over the threshold, thereby triggering some kind of state change (e.g. the door opens); and the avatar's exit from the threshold then triggers a further state change (e.g. the door reverts to its closed state). While this may be useful for something like creating an automatic door, it is not useful for triggering the release to the user of information about the room being entered, as the state of the door (e.g. open or closed) on its own does not tell the system if the avatar passed from one room to another, or if they stopped and moved back into the original room. Such problems are further exacerbated when dealing with multiple avatars within the same world.

If such problems are anticipated and managed well, then the problem of "tarry" time within a zone can be used for temporal passive interaction, that is to say that the longer the avatar "tarries" (i.e. remains) within a zone, then it can be assumed that the user has a greater interest in that area. This assumption can be used to "push" more information to the user, based on the principle that they want to know more because they are spending more time at the location. (The same principle can be applied to the provision of higher levels of graphical detail, with more detailed geometry and higher image resolution provided the longer an avatar remains in an area.)

This kind of temporal interaction is a relatively high-risk strategy, however, as the designer must second-guess the user's reason for lingering in an area, which may be for a variety of reasons (from interest or engagement with the content or with other users, to real-world interruptions such as phone calls and coffee breaks), and must also design a structured method for the phased delivery of information over a defined tarry period. It is strongly recommended that, if temporal interaction is employed, the user is kept informed of how much information is remaining so that the user is not left waiting for more information where none may exist.
Passive interaction strategies are, generally speaking, hidden from the user, being activated only as avatars explore the virtual world; they run in the background and "magically appear" when trigger conditions are met. While this method of interaction can be a highly useful tool for the designer, the hidden nature of these techniques may inhibit the user's discovery of functionality if they do not explore the world fully, or if the trigger areas are in non-obvious locations. Certain techniques can be employed in order to facilitate better user awareness, to encourage the user into certain zones, and to highlight potentially interactive elements to the user.
Possibly the most intuitive, passive method of attracting the user's attention to an area is the use of *signposts*. These can be literal or metaphorical, depending on the nature of the application being designed.

Literal signposting utilizes objects in world that are not part of (or would be considered out of place in) the real-world equivalent of the virtual world. Examples of these would include floating information boards, text or icons that hover over interactive areas, route markers (such as arrows on the floor/walls/posts), or any discordant object and/or special effect which unambiguously can be identified as non-native to the represented environment. The Figure 3.8 shows an example.

Figure 3.8 sign post in the game *Team Fortress 2*.

The principal consequence of using literal signposts is that, by creating visual artefacts that disrupt the perceived autonomy and integrity of the virtual world, they forcefully foreground the artificial nature of the environment, compromising the user's capacity imaginatively and emotionally to experience a sense of presence within the fictional world.

Having said that, such methods may offer a highly efficient means of providing information. Developers must therefore consider what the primary intended purpose of each virtual world is and how far imaginative immersion is necessary or useful to fulfilling that purpose. Consider, for example, a virtual environment, which is designed as a means of scientifically annotating, in three-dimensions, a complex, archaeological site, and to which the "look and feel" of the site is of relatively little importance. In such a case, literal signposting may be an ideal solution. The main drawback is that the visual intrusiveness of such signposting makes it more difficult for such worlds to fulfil other (secondary or tertiary) purposes, such as to evoke the visual appearance of a site. Mitigating the limitations of literal signposting, developers may make their appearance conditional, for example allowing users to toggle literal signposts on or off, or programming them to become visible to a user only to member of a defined user group.

Metaphorical techniques are designed to be more subtle in function and/or form. They may, for example, be used to suggest or promote certain routes, using visual cues that integrate harmoniously with the represented environment. Examples of metaphorical signposts include: the use of sound; changes in light (in the seminal video game Doom, for instance, accessible doors were lit and locked doors were identified by stuttering lighting, while inaccessible doorways, i.e. those only for decoration, remained unlit); themed objects (for example, the use of vases or statues in a representation of an ancient classical building to indicate interactive areas, while not historically accurate, would be in keeping with the theme of the represented world); or the subtle use of special effects such as softly glowing objects or particle systems (dust swirls, blowing leaves, smoke and even clouds of insects have all been used).

Generally speaking, it is good practice to expose users to the chosen signposting method early on in the experience in order to teach the user how to recognise and to use them, and to enable users more naturally to enfold these conventions within their "suspension of disbelief"; indeed, users will often quickly learn to welcome them and the functionality they can release.

User customisation of the platform, or hardware capabilities, may make the implementation of signposting challenging. A system relying on intensity of localised sound (also known as "3D sound") is of little use where the required audio hardware – binaural speakers – is not present, or where the user has chosen to mute sound. Similarly, low-performance graphics cards may not have the capacity to render local lighting or certain kinds of special effects. When designing virtual environments for installations within a venue, audio – whether triggered effect or video sound track – is especially problematic; depending on the acoustics of the building, noise policies (in a library for instance) or appropriateness (such as in a place of remembrance or of worship), the aural aspects of the interface and interaction may disturb other visitors.

3.3.2.2.2 Active Strategies

Whereas passive interaction (even where signposting is used) requires only that the user move around the virtual world, active interaction requires the user consciously to interact with object within the virtual world. The most ubiquitous form of active interaction is "touching" objects within the world through selection, usually the pointing device; indeed, it is a testament to the early developers that this metaphor has become so ingrained in the popular psychology of the computer literate, and feels so "natural", that novel (and often more ostensibly 'intuitive') methods of interaction seem alien and may even work against the user's suspension of disbelief.

Active interaction requires the user to be able to identity the objects which can be interacted with; signposting can be helpful in achieving this. As these points of interactivity are "active", that is to say, they are

programmed to respond to "touch", they lend themselves to a wider range of affordances than is commonly used with the "passive" methods discussed above.
One of the more common identification methods used to alert the user that interaction is possible is a change in the graphical appearance of the cursor / pointing device's onscreen icon from, for example, an arrow to a hand. The method used to trigger an action relies on the platform's capacity to detect this "mouse over" event; depending on its implementation, this can impose high computational requirements, as the position of the cursor in three-dimensional space must constantly be calculated (see earlier notes for the selection of in-world objects for additional problems that may be encountered). Where a cursor is not visible, or in some platforms (such as first-person experiences) the selection of objects is problematic for other reasons, solutions for indicating interactive objects include: locating and signposting target objects that appear only within fixed "gun sights"; highlighting interactive objects that fall within a defined range of the user's avatar; or using the keyboard to tab though a visual index of interactive items on screen.

Most virtual environment platforms will permit the designer to deploy these different approaches in variety of combinations, and it is up to the designer to craft the interaction interfaces necessary for their particular application. As always, consistency in the design of interactive elements is essential to maintaining user's capacity to feel present in the virtual world.

For example, the Theatron3 virtual world, developed on the Second Life® platform and soon to be migrated to Open Simulator, uses passive location tracking tied to a Heads-Up-Display (HUD) to ascertain avatar location within the simulation. Each HUD can be loaded with different trigger zone details to personalise the user experience (allowing one virtual environment to be simultaneously used by many different avatars who have different interests, aims and levels of expertise and skill. In this example, the HUD itself is the primary method of delivering ancillary information, and can be configured in a number of different ways to suit the user's preferred method of receiving data. Local interactions, such as the opening of doors, use collision methods to automate the process, which the user quickly becomes accustomed to (e.g. users rapidly come to understand that only doors that automatically open lead onto further space or content to be explored). Theatron3 does not use signpost techniques, however the HUD does have a mode that enables avatars to move between zones, as well as a "hint" mode that tells the user where his/her avatar is in relation to the selected viewpoint. Active objects are identified through the use of the in-built affordances of pointer icon change and life-like interactive objects such as light-switches (e.g. lights will turn on when a light switch is activated, rather than when a light is touched).

3.3.3 Design Guidelines for System Control

The building blocks of regular desktop applications are often UI elements such as pull-down menus, toolboxes and buttons. These elements are examples of so-called system control techniques, allowing the user to execute commands, change parameters etc. Designing system control for a 3D UI is not trivial: Simply adapting 2D desktop-based UI elements is seldom the best solution. Unfortunately, scarcely any research has been performed in this area. Nevertheless, Bowman et al (2004) have formulated a set of guidelines for 3D UI system control based on anecdotal evidence and experience:

- **Avoid disturbing the flow of action of an interaction task.** *Example Jenny effortlessly activates a pie menu for choosing different rendering styles by pushing the B button of the Wii mote.*
- **Prevent unnecessary changes of the focus of attention.** *Example Martin gets really disturbed that the settings menu covers the whole screen when he's interacting with the* Flaminia Reloaded Virtual Museum.
- **Avoid mode errors.** *Example Isabella gets clear visual and auditory feedback that she has changed to system control mode.*
- **Use an appropriate spatial reference frame.** *Example Matt always has the tools menu in his field of view since the* Etruscan Virtual Museum *is tracking his head movements.*
- **Structure the functions in an application.** *Example Marco can choose from 50 different stories in a cascade menu inside the* Mercati di Traiano Virtual Museum*.*
- **Consider multimodal input.** *Example John points with his index finger at a clay figurine in the Cyprus Museum Virtual Museum and says "Show meta-data".*

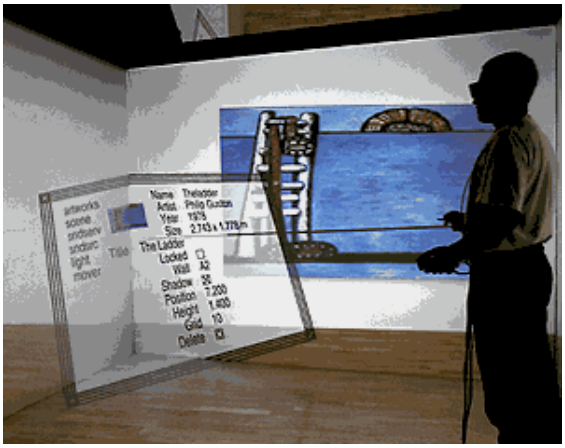


Figure 3.9 System control in an immersive Virtual Museum.
Image courtey of Gerhard Eckel, Fraunhofer IMK.

Furthermore, the know-how and experience from the videogame industry and the virtual world community is a valuable source of knowledge. In this context, the primacy of the screen as the main output means that the developer should try to maximise the amount of the screen occupied by the actual virtual world (also known as maximising "screen real estate") and, correspondingly, to minimise the visible impact of on-screen UI elements.

From the purist view of virtual reality (and increasingly video games) (Wilson, 2006) the ideal would be the total absence of any persistent interface elements, leaving only the world itself visible to the user. Virtual worlds in the cultural heritage sphere, however, while they share similar concepts with VR, are often designed to facilitate shared, collaborative experiences, and these complex interactions with the world and with other users require the user to have access to a greater range of functionality.

As a rule of thumb, as far as possible, all non-essential UI elements should be removed from the persistent, on-screen interface, while providing methods enabling the user to access them when needed. There will inevitably be a compromise between provision of an uncluttered view of the world through the screen interface, on the one hand, and intuitive access to information/functionality, on the other. Wherever possible, the designer should ensure that these two aspects are graphically harmonised and that they also have redundant access functionality (i.e. keyboard shortcuts), thereby diminishing the perceived intrusiveness of the user interface structure into the user's awareness.

Unless the user interface and its interaction is being created for a specific screen, such as in a museum installation, attention must be paid to ensure that all interface components can be scaled appropriately to match the screen size and resolution available to the end-user. Failure to do this can lead to UI elements overcrowding or overflowing the screen, preventing access to required functions, or excessively small, making interface elements difficult or impossible properly to perceive.

When UI elements of 3D UI:s in the cultural heritage domain closely resemble those that users are likely to have encountered in other contexts – such as through using commercial games platforms – then users are likely to understand and to become comfortable with them more quickly, and this will, in turn, help to maximise uptake. An example of such a UI element is the so-called pie menu, which is widely used in videogames to reduce both screen clutter and scaling issues. As such a menu is initiated by the user, concerns over modality are simpler to address and the grouping of items in a central point makes ergonomic and efficient workflow possible. Moreover the interactive region can be extended radially out from the pie centre to take advantage of the full height and width of the screen, tying the pointing/selection method to a screen quadrant rather than a specified target, and thus reducing problems associated with the scaling of interface components. Except for these

technical advantages, pie menus also have very good usability. They are faster and more reliable to select from than linear menus, because selection depends on direction instead of distance. Furthermore, experienced users can use muscle memory without looking at the menu while selecting from it.

Another UI element ofter seen in videogames is the Heads-Up Display (HUD), taken from the projected display originally developed for military aviation that allowed pilots to view information without having to look away from their usual viewpoints. HUDs can include both informational display as well as providing an additional user interfaces giving access to functionality through the kinds of interaction methods described previously. Designers should adhere to good practice regarding both ergonomic layout and use of iconography. As the HUD will take up screen "real estate", designers should if possible include two core functions: the first allows the user to change the HUD's level of transparency and the second is a "hide" or "minimize" function, reducing the HUD to a discreet, non-intrusive onscreen presence (Figure 3.10). Like other UI elements, HUDs should be scalable to ensure that the variety of possible configurations of screen size and resolution is accommodated. Where possible the user should be able to relocate the HUD within their screen space, to suit their own viewing/ergonomic requirements, but the design should prevent the HUD from partially or fully positioned outside the viewing area, and so becoming "lost". If allowed by the software platform, the HUD should assist user interaction by including additional redundancy, and may also provide "tool tip" information when the pointing device rolls over HUD items. Where the HUD is using words, rather than pure iconography, designers should take into consideration localization issues, ensuring that the language employed is consistent.

Figure 3.10 The Heads-Up-Display (HUD) of the game *Metroid Prime* is easily accessible, necessary, unobtrusive and also completely customizable as to be invisible if the player wishes.

A different approach is to remove the HUD partially or completely and instead integrate the information into the virtual environment of the game. The motivation is to increase realism and hence the sense of presence experienced by the player. A recent example could be seen in the game *Dead Space* (Figure 3.11).

Figure 3.11 The game *Dead Space* has no traditional HUD. Notice the health bar on the back pack of the player's avatar.

4 Deploying 3D Graphics to the World Wide Web

4.1 Overview

The delivery of 3D content through the World Wide Web comes with a considerable delay with respect to other digital media such as text, still images, videos and sound. Just like it already happened for commodity platforms, 3D Computer Graphics is the latest of the abilities acquired by the web browsers. The main reason for this delay is likely the higher requirements for 3D graphics in terms of computational power. However, nowadays sophisticated rendering techniques can be implemented thanks to the ability of modern GPUs to perform complex tasks and the possibility to interact with the graphics system directly within web pages, even using relatively slow interpreted languages. In this section we provide an overview of the visualization technologies related with the development of graphics application for the Web.

Before the introduction of advanced 3D graphics capabilities inside the Web Browsers, several technologies have been developed over the years to overcome such limitation. The *Virtual Reality Modeling Language (VRML)* [VRML] (then superseded by *X3D* [X3D]) was proposed as a text based format for specifying 3D scenes in terms of geometry and material properties, while for the rendering in the web browser it is required the installation of a platform specific plug-in. *Java Applets* are probably the most practiced method to add custom software components, not necessarily 3D, in a web browser. The philosophy of Java applets is that the URL to the applet and its data are put in the HTML page and then executed by a third part component, the Java Virtual Machine. The implementation of the JVirtual Museum on all the operating systems made Java applets ubiquitous and the introduction of binding to OpenGL such as JOGL [JOGL] added control on the 3D graphics hardware. A similar idea lies behind the ActiveX [ACTIVEX] technology, developed by Microsoft since 1996. Unlike Java Applets, ActiveX controls are not bytecode but dynamic linked Windows libraries which share the same memory space as the calling process (i.e. the browser), and so they are much faster to execute. These technologies allow to incorporate 3D graphics in a web page but they all do it by handling a special element of the page itself with a third party component. More recently, Google started the development of a 3D graphics engine named *O3D* [O3D]. O3D is also deployed as a plug-in for browsers, but instead of a black box, non-programmable control, it integrates into the browser itself, extending its JavaScript with 3D graphics capabilities relying both on OpenGL and DirectX. O3D is scene graph-based and supplies utilities for loading 3D scenes in several commonly used formats. The turning point for 3D graphics and the Web is represented by the

introduction of *WebGL* [WEBGL], an API specification produced by the *Khronos Group* [KHRONOSGROUP], that, as the name suggests, defines the JavaScript analogous of the OpenGL API for C++. WebGL closely matches OpenGL ES 2.0 and, most important, uses GLSL as the language for shader programs, which means that the shader core of existent applications can be reused for their JavaScript/WebGL version. Since WebGL is a specification, it is up to the web browsers developer to implement it. At the time of the writing of this document WebGL is supported by Firefox, Chrome, and Safari. A number of JavaScript libraries are being developed to provide higher level functionalities to create 3D graphics applications, one of the most famous one is *three.js* [THREEJS].

For example *WebGLU* [WEBGLU], which is the WebGL correspondent of *GLU* [GLU], provides wrappings for placing the camera in the scene or for creating simple geometric primitives, other libraries such as *GLGE* [GLGE] or *SceneJS* [SceneJS] uses WebGL for implementing a scene graph based rendering and animation engines. In the following we give a detailed description of the three visualization technologies that we will be the basis for the development of the web-based visualization tools of the CIF, i.e. *X3DOM*, *OSG4Web* and *SpiderGL*. First we present the plugin-based solution, OSG4WEB, then the browser embedded solutions, i.e. X3DOM and SpideGL.

4.2 OSG4Web - a plugin-based visualization technology for the WWW

4.2.1 Technical description

OSG4Web is a cross browse and cross platform plugin that wraps some of the advanced viewing and navigating functionality of one general purpose Scene Graph rendering libraries, *OpenSceneGraph*, that is one of the most used open source solutions for visual simulation. This plugins use the underlying rendering middleware for allowing a rendering window to be embedded within a generic web page allowing the user to navigate and interact with a scene defined by the data nodes that are dynamically loaded using standard http protocols. The plugin application is a generic navigator, allowing some predefined interaction with specifically tagged objects (nodes) currently present in the scene.

By layering on OpenSceneGraph we are also leveraging many tools that have been grown around this library: one of the most useful tool is an open source terrain generator (Virtual Planet Builder) that allow to postprocess large GIS landscape data into tiled and paged terrain hierarchies ready to be viewed on line.

The scene graph approach let a complex scene, comprised of several components such as terrain, vegetation, low poly urban building as well as higly detailed reconstruction to be (almost) seamlessly integrated into a single scene that can be navigated by the same viewer application.

The plug-in consist of:

1. A library of common functionalities such as scene navigators, scene interactors, picking tools, etc. These are needed extensions to the base OpenSceneGraph library.
2. A flexible application implementing basic scene interaction, allowing different projects to be defined by scene data, configuration parameters and embedded page configuration.
3. A small browser embedding layer that abstract the different browser and operating system diffrencies: OpenSceneGraph itself is completely platform independent and almost windowing agnostic but nevertheless the different platforms and browser require adaptation (for example on Windows and Linux the rendering happens in a separate process, while on Mac OSX this is not permitted).

The application is designed to maintain a strict decoupling between rendering engine code and browser code. Regarding deployment, both Firefox extension framework (that provides upgrading out of the box) as well as standalone installers have been tested extensively.
On the Server side requirements, the data exchanged are in the native OpenSceneGraph format to allow for best efficiency and are retrieved by the Plugin OpenSceneGraph component through plain http requests. The terrain database is generated in a batch preprocessing phase, then published as an http folder. The dynamic information, such as viewpoints, paths, model placements and 3d hyperlinks are entered with a CMS-like simple back office interface and are kept in a MySQL database.
As an example of application of this technology, in the next sub-section we present briefly the Aquae Patavinae project (http://www.aquaepatavinae.lettere.unipd.it/portale/?page_id=2174). Further details about this plugin and related projects can be found in [Calori2009, Fanin2011].

4.2.2 The Aquae Patavianae Project

This open-source virtual archaeology project (shown in Figure 4.1) presents an approach for the reconstruction of archaeological contexts and its implementation for online 3D exploration using a web browser plug-in with support for large 3D landscapes and advanced interactions, applied also to virtual museums domains. *Aquae Patavinae project* aims at enriching cultural knowledge and dissemination of the great Roman archaeological discoveries of Montegrotto, through 3D visualization and interaction. A multi-resolution and multi-scale virtual world exploration is possible thanks to natural interface integration, such as touch screens, low-cost camera-tracking and multi-touch gestures. The purpose is to simplify interactive exploration and let users to focus on valuable hotspots across the landscape, with related information and improved immersivity. Up to now, the archaeological sites of the territory remained unknown and underestimated: damaged remains are often inaccessible and appear as too fragmented to enable a proper interpretation without archaeological skills. This

online graphics application can be an important tool to help visitors to understand the archaeological landscape and its potential aspect in the past.

Figure 4.1 Aquae Patavinae Project. A screenshot during an online navigation session.

4.3 X3D and X3DOM

4.3.1 Introduction and Objectives

Besides the aforementioned browser plugins, Java3D (Sun, 2007) – a scene-graph system that incorporates the VRML/X3D (Web3D, 2008) design – was one of the first means for 3D in the browser. However, it never really was utilized for the web and today Java3D is no longer supported by Sun at all. The open ISO standard X3D in contrast provides a portable format and runtime for developing interactive 3D applications. X3D evolved from the old VRML standard, describes an abstract functional behaviour of time-based, interactive 3D multimedia information, and provides lightweight components for storage, retrieval and playback of real-time 3D graphics content that can be embedded into any application (Web3D, 2008). The geometric and graphical properties of a scene as well as its behaviour are described by a scene-graph (Akenine-Möller et al., 2008). Since X3D is based on a declarative document-based design, it allows defining scene description and runtime behaviour by simply editing XML without the need for dealing with low-level C/C++ graphics APIs, which not only is of great

importance for efficient application development but also directly allows its integration into a standard web page. Further, using X3D means that all data are easily distributable and sharable to others. Despite proprietary rendering systems that all implement their own runtime behaviour, X3D allows *developing portable 3D applications*.

The X3D specification (Web3D, 2008) includes various internal and external APIs and has a web-browser integration model, which allows running plugins inside a browser. Hence, there exist several X3D players available as standalone software or as browser plugin. The web browser holds the X3D scene internally and the application developer can update and control the content using the Scene Access Interface (SAI), which is part of the standard and already defines an integration model for DOM nodes as part of SAI, though there is currently no update or synchronization mechanism. To alleviate these issues, with the X3DOM framework (Behr et al., 2009) a DOM-based integration model for X3D and HTML5 was presented to allow for a seamless integration of interactive 3D content into HTML pages. The current implementation is mainly based on WebGL (Khronos, 2010), but the architecture also proposes a fallback model to allow for more powerful rendering backends, too (Behr et al., 2010), which will be explained in the next section.

To overcome the old plugin-model, Khronos promotes WebGL as one solution for hardware accelerated 3D rendering in the web. The imperative WebGL API (WebGL, 2010) is a JavaScript (Crockford, 2008) binding for OpenGL ES 2.0 (Munshi et al., 2009) that runs inside a web browser, thereby allowing for native 3D in the web. Most browsers like Mozilla, Apple WebKit, Google Chrome and Opera (except Microsoft"s IE) followed with WebGL-enabled developer builds. By utilizing OpenGL ES 2.0 as basis, it was possible to define the WebGL specification in a platform independent manner, since on the one hand OpenGL 2.1 (the current standard for desktop machines) is a superset of ES 2.0. And on the other hand, most recent smartphones, already have chips being conformant to that standard.

WebGL (WebGL, 2010) describes an additional 3D rendering context for the HTML5 <canvas> element (W3C, 2009a) by exposing the rendering API via new JavaScript objects and methods acting on the canvas object. The 3D rendering context is then acquired via gl = canvas.getContext('webgl'). If the returned gl object is defined and not null, the web browser supports WebGL – in this case the gl object provides all API calls. In contrast to standard desktop OpenGL (Shreiner et al., 2006) Open GL ES 2.0 has no support for the old fixed function pipeline (i.e., no matrix stack etc.) but is instead completely based on GLSL shaders (Rost, 2006). Another drawback is the fact that the web-developer has to deal with low-level graphics concepts (maths, GLSL-shaders, attribute binding, and so on). Moreover, JavaScript scene housekeeping can soon lead to performance issues, and there is still no uniform notion of metadata or semantics for the content possible.

During the last year, WebGL-based libraries such as WebGLU (DeLillo, 2009), which mimics the old OpenGL fixed-function pipeline by providing appropriate concepts, emerged as well as rendering frameworks building on top of WebGL by providing a JavaScript-based API. For instance GLGE (Brunt, 2010) is a scene-graph system

that masks the low-level graphics API calls of WebGL by providing a procedural programming interface. Likewise, SpiderGL (see Section 4.4) provides algorithms for 3D graphics, but on a lower level of abstraction and without special structures like the scene-graph. These libraries are comparable to typical graphics engines as well as to other JavaScript libraries like jQuery (cp. http://jquery.com/), but none of them seamlessly integrates the 3D content into the web page in a declarative way nor do they connect the HTML DOM tree to the 3D content. In this regard, the aforementioned jQuery aims at simplifying HTML document traversing, event handling, and Ajax interactions, thereby easing the development of interactive web applications in general. However, using libraries like SpiderGL forces the web developer to learn new APIs as well as graphics concepts. But when considering that the **Document Object Model** (**DOM**) of a web page already is a declarative 2D scene-graph of the web page, it seems natural to directly utilize and extend the well-known DOM as scene-graph and API also for 3D content.

4.3.2 Declarative 3D Content on the Web

In order to bring 3D content in a declarative way into web browser front end, specific coupling mechanism have to be considered. This relates to an adequate JavaScript layer as well as interaction and event handling mechanism that need to be available as enabling building blocks for declarative content. The following sections provide an overview of the required building blocks in order to establish fully fledged 3D enhanced web sites.

4.3.2.1 DOM Integration

X3DOM integrates 3D content into the browser without the need to forge new concepts, but utilizes today's web standards and techniques, namely HTML, CSS, Ajax, JavaScript and DOM scripting. Figure 4.2 shows a simple example, where a 3D box is embedded into the 2D DOMtree using X3DOM. Though HTML allows declarative content description already for years, this is currently only possible for textual and 2D multimedia information.

Hence, the goal is to have a declarative, open, and human-readable 3D scene-graph embedded in the HTML DOM, which extends the well-known DOM interfaces only where necessary, and which thereby allows the application developer to access and manipulate the 3D content by only adding, removing or changing the DOM elements via standard DOM scripting – just as it is nowadays done with standard HTML elements like , , <img> or <canvas> and their corresponding CSS styles. Thus, no specific plugins or plugin interfaces like the SAI (Web3D, 2009) are needed, since the well-known and excellently documented JavaScript and DOM infrastructure are utilized for declarative content design. Obviously, this seamless integration of 3D contents in the web browser integrates well with common web techniques such as DHTML and Ajax. Furthermore, semantics integration can be achieved with the help of the X3D metadata concept for creating mash-ups (i.e. a recombination of existing contents) and the like or for being able to index and search 3D content. Hence, the

goal is to have a declarative, open, and human-readable 3D scene-graph embedded in the HTML DOM, which extends the well-known DOM interfaces only where necessary, and which thereby allows the application developer to access and manipulate the 3D content by only adding, removing or changing the DOM elements via standard DOM scripting – just as it is nowadays done with standard HTML elements like , , <img> or <canvas> and their corresponding CSS styles.

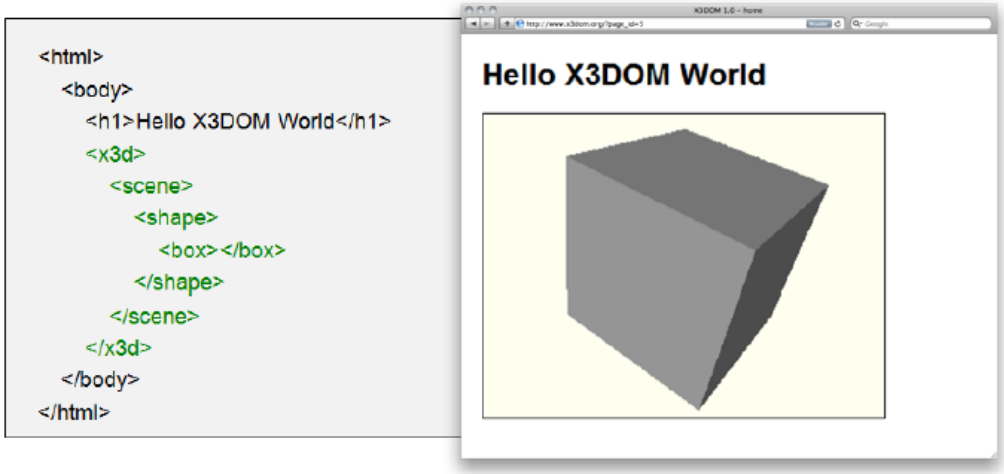


Figure 4.2 Simple Hello World object as declarative embedding of 3D content in a web browser (here Google Chrome, through WebGl)

Thus, no specific plugins or plugin interfaces like the SAI are needed, since the JavaScript and DOM infrastructure are utilized for declarative content design. Obviously, this seamless integration of 3D contents in the web browser integrates well with common web techniques such as DHTML and Ajax. Furthermore, semantics integration can be achieved with the help of the X3D metadata concept for creating mash-ups (i.e. a recombination of existing contents) and the like or for being able to index and search 3D content.

4.3.2.2 Interaction and Events

Most visible HTML tags can react to mouse events, if an event handler was registered. The latter is implemented either by adding a handler function via element.addEventListener() or by directly assigning it to the attribute that denotes the event type, e.g. onclick. Standard HTML mouse events like "onclick", "onmouseover", or "onmousemove" are also supported for 3D objects alike. Within the X3DOM system we also propose to create a new 3DPickEvent type, which extends the W3C MouseEvent IDL interface (W3C, 2000) to better support 3D interaction. Dedicated interaction concepts required by Virtual Museum on the web as identified so far are:

- picking,
- rotation,
- zoom (Object Eploration)
- camera based navigation (Dynamic (Walkthrough) Scenarios)

4.3.2.3 Animations

There are several possibilities to animate virtual objects (e.g. for showing an ancient device in action etc.), ranging from updating attributes in a script every frame over standard X3D interpolator nodes up to using CSS-3D-Transforms und CSS-Animations, which are currently given as W3C working draft and only implemented in WebKit based web-browsers such as Apple Safari and Google Chrome. While X3D interpolators are supported by current Digital Content Creation (DCC) tools – an important point when processing the raw data and exporting to other formats – and are also able to animate vertex data (e.g. coordinates or colors), CSS animations are easily accessible using standard web techniques.

4.3.3 Application Examples

The following section provides some concepts for potential use of declarative 3D content for Virtual Museum installations on the web establishing an online experience.

4.3.3.1 Object Exploration

One of the most basic use cases one can think of here is the examination of individual objects of the virtual heritage. In a typical scenario the 3D object is presented to the user such that he or she can examine it from all directions by simply moving and rotating it (or the virtual camera respectively) around with the mouse or a similar device. Concerning visualization this is a rather simple scenario in that the 3D scene itself keeps static. Here, Figure 4.3 shows some screenshots of the web-based visualization of Cultural Heritage objects provided by the 3D COFORM consortium.

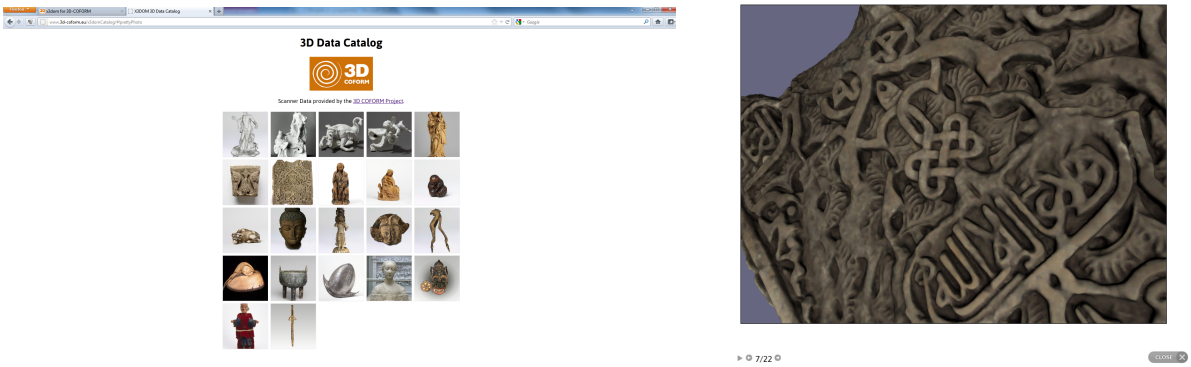


Figure 4.3 (Left) A screenshot of the digital catalogue. (Right) An online navigation session.

As can be seen, all geometric 3D objects are visualized in the web-browser by simply utilizing our open-source X3DOM framework for rendering the 3D content in real-time. This is especially notable in that this is still almost raw data stemming from 3D Laser scans, which is neither reduced nor somehow otherwise prepared for real-time rendering.

Additionally, by extending the web page with some standard JavaScript code for DOM scripting – where appropriate – the user can also interactively manipulate the data using standard 2D GUI elements (e.g. buttons and sliders) as for instance provided by the aforementioned JavaScript library jQuery. This can be useful to vertically or horizontally translate a clipping plane in order to cut away stratigraphic sequences and the like. Furthermore, it is also possible to allow the user to directly interact with an object by clicking on a certain point of interest etc., which then for instance triggers a popup HTML element containing some additional information.

4.3.3.2 Dynamic (Walkthrough) Scenarios

Other possible scenarios in CH embrace walkthrough worlds and the inspection of larger models like ancient city models and similar territories in virtual archaeology. With the Cathedral of Siena (c.f. Figure 4.4) a classical guided walkthrough scenario is described in (Behr et al., 2001). Generally, floor plans (Figure 4.4, right) are a commonly used metaphor for navigation. This is for two reasons: for one thing the plan allows the user to build a mental model of the environment, and for another it prevents him from getting lost in 3D space. In this regard, camera paths with predefined animations are another frequently used means for guided navigation. In X3DOM camera animations can be easily accomplished by using one of the aforementioned animation methods, like for instance X3D interpolators.

Figure 4.4 Example of the Digital Cathedral of Siena (cf. Behr et al., 2001): the left image shows the rendered 3D view of the cathedral's interior and a virtual guide, and the right image shows the 2D user interface in order to "instruct" the avatar to move to the clicked position

Alternatively, the scene author can only define some interesting views and let the system interpolate between them. The resulting animations are automatically generated if one binds the camera, e.g. when switching between different Viewpoint nodes (or cameras), which are part of the content. The same method is also used to calculate the animation-path if the current view is being reset or if the current camera-view shall be moved to the „show all" position. As explained, it is furthermore possible to freely navigate within the 3D scene in order to closely examine all geometric objects. Besides this, the user can also walk or fly through e.g. a reconstructed city model or an old building as shown in Figure 4.5. Like every X3D runtime, also the current WebGL-/ JS-based implementation of X3DOM provides some generic interaction and navigation methods. As already outlined, interactive objects are handled by HTML-like events, while navigation can either be user-defined or controlled using specific predefined modes. Therefore, we added all standard X3D navigation modes, i.e. "examine", "walk", "fly" and "lookAt". The content creator is free to activate them, for instance directly in the X3D(OM) code with <navigationInfo type='walk'>, or to alternatively write his own application-specific navigation code. In the WebGL-based implementation the modes use the fast picking code (required for checking front and floor collisions) based on rendering the required information into a helper buffer, which performs well even for larger worlds. However, in order to realize large model visualizations additional streaming and encoding mechanism have to be implemented. Thus, users might also be able to visualize large scale terrain models used throughout virtual archaeology.

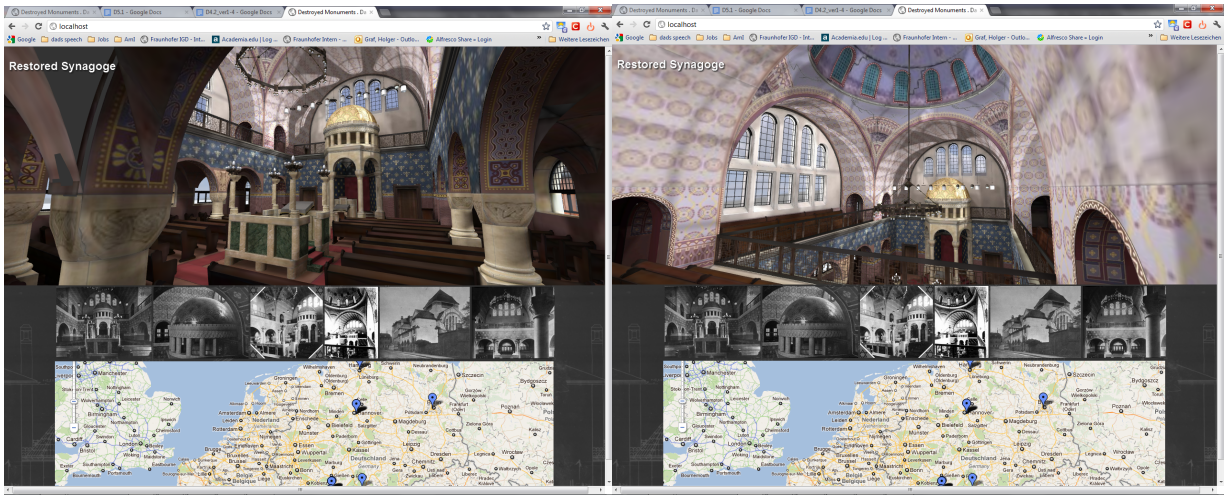


Figure 4.5 Online Virtual Museum visualizing the destroyed synagogue; exemplary implementation of camera fly through mode attached to the "on click" mouse event.

4.3.3.3 Mobile micro Virtual Museum - X3D Portability and Scalability

AR as a rapidly emerging technology combined with the ubiquitous computing power of modern mobile devices means having the desired information in one pocket. With the help of the video-see-through effect the information – such as 2D images or the 3D reconstruction of digitally rebuilt information can be superimposed onto the video image, or the real world respectively, by using computer-vision-based tracking techniques.

In this context the term Mixed Reality means to be able to bring together (web-) content and location-based information directly on site. Especially when producing content for (mobile) MR applications, the unification of 2D and 3D media development is an essential aspect. Other important factors for authoring and rapid application development are declarative content description, flexible content in general (not only for the cultural heritage domain, but also for the industry etc.), and interoperability – i.e., write once, run anywhere (web/ desktop/ mobile). In X3DOM we want to achieve this by utilizing JavaScript and DOM infrastructure also for 3D in order to bring together both, open architectures and declarative content known from web design. Whole Virtual Museum applications could be established as Apps and made accessible through recent app stores for a broader community. The possibility to app-independent visualizations furthermore enables context sensitive and on-demand information retrieval, which is even more of interest for distributed content development using available web standards. But when limiting oneself to the pure WebGL-based JS layer of X3DOM, at the moment special apps for handling the tracking part are still needed (e.g. by using Flash or the InstantPlayer

plugin), because access to the camera image data is required but not yet supported in HTML5. However, we envisage to propose a <device> tag within the run of V-MUST in order to allow for micro museums in the near future.

4.4 SpiderGL

4.4.1 Introduction and objectives

Thanks to the combination of hardware and software capabilities and performances, coupled with a high-speed data channel, it is nowadays possible to effectively and natively handle real-time 3D graphics within web pages. In particular, by exploiting the asynchronous features provided by the runtime environment of the web browser, it is possible to manage large datasets in a natural out-of-core fashion. The creation of fast and reliable visualization algorithms that allow the user to explore huge environments (like Google Earth and Bing Maps) implies that multiresolution algorithms should be developed with network streaming in mind, both in terms of caching mechanisms and the actual representation of a data packet. Alongside, it is easy to see how the new WebGL 3D technology will bring closer web developers, which are more and more interested in learning 3D graphics and CG developers, which will try to deploy their algorithms to less powerful platforms. The question is now what still separates a standalone graphics application developed in C++ or other programming language and from a JavaScript graphics application for the Web.. One obvious answer is execution speed, but there are other gaps to be filled, such as:

- **Asynchronous content loading:** many CG algorithms, especially when dealing with multiresolution datasets, make intensive use of multi-threading for asynchronous (down)loading of textures or geometry data from different cache levels. This is vital to avoid the application to freeze while waiting for a texture to be loaded from RAM, disk or even a remote database to GPU. On the other hand JavaScript still does not officially support mult-ithreaded execution.
- **Shape data loading from files:** there are many le formats for 3D models and as many C++ libraries to load them [VCGLIB, OPENMESH, CGAL]. JavaScript includes a series of predefined types of objects for which the standard language bindings expose native loading facilities (i.e. the Image object), but such bindings for 3D models have yet to come.
- **Math:** linear algebra algorithms for 3D points and vectors are very common tools for the CG developer, and a large set of dedicated libraries exists for C++ and other languages. Although many JavaScript demos for mathematical algorithms can be found just browsing the web, a structured library with the specific set of operations used in CG is still missing.

- **WebGL wrapping:** the WebGL specification is very similar to OpenGL ES 2.0, which means that there are significant changes w.r.t. OpenGL, for example there are no matrix or attribute stacks and there is no immediate mode. Although these choices comply to the bare-bones philosophy of OpenGL ES 2.0, they also imply incompatibility even with OpenGL 3.0, which, for example, still provides matrix stack operations.

SpiderGL is explicitly design to fill the gaps mentioned. It extends JavaScript by including geometric data structures and algorithms and wraps their implementation towards WebGL. In particular, SpiderGL was designed keeping in mind three fundamental qualities:

- **Efficiency:** With JavaScript and WebGL, efficiency is not only a matter of asymptotic bounds on the algorithms, but the ability to and the most efficient mechanism to implement, for example, asynchronous loading or parameters passing to the shader programs, without burdening the CPU with respect to a bare bone implementation;

- **Simplicity and Short Learning Time:** Users should be able to reuse as much as possible of their former knowledge on the subject and take advantage of the library quickly. For this reason SpiderGL carefully avoids over-abstraction: almost all of the function names in SpiderGL have a one to one correspondence with either OpenGL or GLU commands (e.g. the SpiderGL function sglLookAt for setting up the camera pose matrix), or with geometric/mathematics entities (e.g. SglSphere3, SglMeshJS).

- **Flexibility:** SpiderGL does not try to hide native WebGL functions, instead it provides higher level functionalities that fulll the most common needs of the CG developer, who can use SpiderGL and WebGL calls almost seamlessly.

4.4.2 Modules

SpiderGL is composed of five modules, distinguished by different color in Figure 4.3:

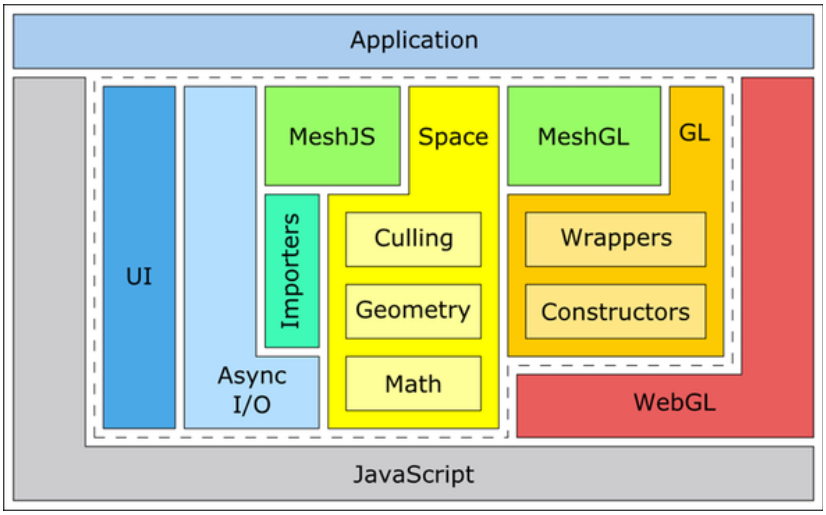


Figure 4.3: SpiderGL Architecture. The library is logically composed of ve modules: MATH (linear algebra and geometry, GL (WebGL wrapping), MESH (polygonal mesh denition and rendering), ASYNC (asynchronous content loading) and UI (event handling and interactors).

A brief description of each module follows:

- **MATH:** Math and Geometry utilities. Linear algebra objects and functions, as well as geometric entities represents the base tools for a CG programmer.
- **GL:** Access to WebGL functionalities. The GL module contains a low-level layer, managing low-level data structures with no associated logic, and a highlevel layer, composed of wrapper objects, plus a series of orthogonal facilities.
- **MESH:** 3D model denition and rendering. This module provides the implementation of a polygonal mesh (SglMeshJS), to allow the user to build and edit 3D models, and its image on the GPU side (SglMeshGL). SpiderGL handles the construction of a SglMeshGL object from a SglMeshJS. ASYNC : Asynchronous Content Loading. Request objects, priority queues and transfer notiers help the programmer to implement the asynchronous loading of data.
- **UI :** User Interface. A GLUT{like framework and a series of typical 3D manipulators allows a quick and easy setup of the web page with 3D viewports and provide effective management of user input.

For the complete insights and technical details of the SpiderGL library we refer to the PhD thesis of his main developer [SPIDERGL].

4.4.3 Applications

In the following we report some applications of the SpiderGL graphics library. As currently stated its main aim is to support actively the development of advanced graphics applications for the web. Some examples of applications of SpiderGL are reported in the next figures (4.4, 4.5 and 4.6), only to give an idea of their possibilities. The applications shown are: large meshes visualization through the Web, online visualization of multi-resolution terrain and remote rendering of polynomial texture maps.

Figure 4.4: Large Meshes Visualization. Vertex index limit is automatically overcome with packed-indexed primitive stream. Here a model of the Michelangelo's David statue with 1M triangles is rendered at about 100 FPS on a web browser. (Left) The whole mesh. (Center) The colored chunks after splitting. (Right) A close-up of the statue head.

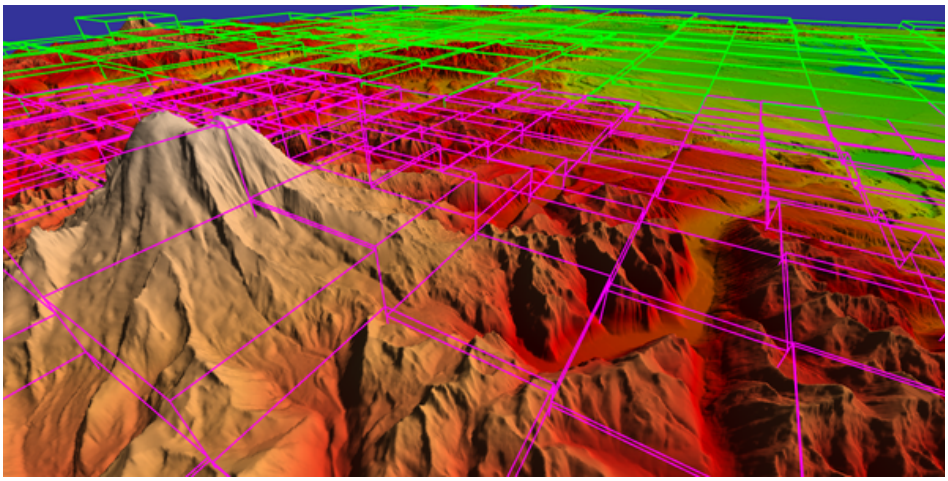


Figure 4.5: Multiresolution Terrain Visualization. A snapshot of an adaptive multiresolution rendering of a large terrain model (Pudget Sound model). Green and magenta boxes represent, respectively, internal and leaf nodes o f a quad-tree used to speed up the data transmission.

Figure 4.6: Polynomial Texture Maps. Two frames captured with different illumination with a detail zoomed for best inspection. Light position is bound to the position of the mouse on the window. Green boxes represents leaf nodes of a quad-tree used to speed up the data transmission.

5 Design of the CIF

Following the above workflows' descriptions, the in-depth discussion about the design of user interfaces and the detailed overview of the 3D visualization technologies for the WWW, we propose an initial design of the *Common Implementation Framework (CIF)*, its integration mechanisms and its profile.

In its core, the CIF will be an ***harmonized pool of tools*** rather a unique implementation framework with adequate interfacing mechanism at different levels:

- Transcoding mechanism ensuring interoperability between tools
- Scripting Interfaces enforcing transmedia authoring
- Services exposing functionality of dedicated tools to the Virtual Museum actors
- Application Templates in order to ensure semi-automatic processes and workflows (Integration at process level)

The CIF will be made available through the V-MUST portal and its platform providing the necessary services and repository infrastructure.
Thus, the foreseen integration mechanism wrap existing tools into services exposing dedicated functionality and lead to the establishment of new services for the realization of online Virtual Museum. Its embedding as web services provides the required framework for common implementation tasks and provides a harmonized tool chain for the presentation and interaction. We expect that the Virtual Museum actors (developers, stakeholders) achieve faster and optimized workflows for the establishment of Virtual Museum and related applications.
As just mentioned, this proposal is strictly connected to the specification of the V-MUST Platform, its service design and related repository (see Deliverable 4.2). The next sections provide an overview of the identified tools requested by the developers and stakeholders and provide a brief description of the tools with its impact on the CIF. It therefore comprises existing tools strictly related to the workflows, and further (new) tools and components that will be part of the CIF even if those are not coming from the workflow analysis. Then, we present some services (mainly web-based) that the CIF will provide. We underline that not all the tools and services here described will be part of the final release of the CIF, but that such tools and services are the needed/requested ones. At the end of this chapter, the integration mechanisms between tools and components of the CIF will be described with a first design of a proposed new architecture exposing the CIF transparently to the end users (Virtual Museum actors).

5.1 Tools and Components coming from the analysis of the workflows

The subsequently presented and selected tools/components are the ones considered important according to the analysis of the partners' workflows plus the needs and requirements coming from the V-MUST technical questionnaire (see deliverable 4.1 for a detailed description of this questionnaire and the related data analysis). These two sources of information allow us to select the following tools/components as the main ones for the development of the CIF. Other tools/components can be integrated further in the CIF in order to support future functionalities/services. As just stated, many of them would be integrated at different levels with the V-MUST repository; we refer it to the Deliverable 4.2 for a complete description of it.

5.1.1 Data format and conversion tools

One of the main feature of the CIF is surely the capability to convert (in a complete automatic way or not depending on the cases) between different multimedia data format in order to increase the interoperability of the tools/components of the CIF. Additionally, conversion tools are required to automatically/manually obtain in the repository an interchange format copy of the multimedia data and/or upload the preferred file format and allow to download or the original one or a real-time converted interchange file format.

For this reason one of the component of the CIF will be a set of conversion tools between multimedia data formats. According to the analysis coming from the deliverable 4.1 some data formats for the 3D models that should be handled are the following:

- 3D Studio Max data format (3DS)
- Blender data format
- COLLADA
- PLY data format
- X3D data format
- Common image data format: JPEG, TIFF, PNG, ..

The first data formats arise as a need due to the fact that 3D Studio Max and Blender emerge as common practice tools. Additionally, the conversion between other 3D data format and the Blender data format is particularly useful especially if remote rendering services will be made available; in fact, a remote rendering farm service should necessarily be Blender-based, for licensing and customization reasons. COLLADA should be supported due to its nature of universal interchange format and wide diffusion. PLY is used by many 3D processing tools, like as Meshlab and many others. The Meshlab server version of Meshlab just support many

conversion between different data format. X3DOM has also conversion functionalities to allow users to put their content on the web in a X3D compatible format.
The conversion between different image formats is another issue that should be taken into account. For this purpose many available Open Source software tools such as ImageMagick or others are just available and can be easily integrated in the CIF using a scripting language.
During the development of the set of conversion tools, a document with conversion guidelines will be also prepared in order to ensure a support for those workflow based on multimedia data that are not comprise in the set of data format supported by the V-MUST platform. This document will include also a description of the limitations of the conversion between different data formats.

5.1.2 Tools for modeling, processing and rendering

Some of the most widely used tools in the field of Virtual Museum development will be made available by the V-MUST platform and integrate as more as possible in the CIF. These tools will provide the basis for modeling, processing and rendering of 3D models. Blender and Meshlab will constitute the main part of such tools. Tools for the processing of GIS data is at the moment not defined, even if tools of this type are a common needs. Both these tools will be integrated in some of the services that will be provided, for example Meshlab may be integrated in the CIF to provide automatic mesh cleaning service, Blender to provide remote rendering, and so on. Other tools that are of interest for some Virtual Museum developers are the Arc3D tool (http://www.arc3d.be) and the VisualSFM (http://www.cs.washington.edu/homes/ccwu/vsfm) tool. This two tools are image-based reconstruction tools, i.e. tools that employ computer vision algorithms to obtain 3D models starting from a collection of images. Arc3D is a fully automatic image-based reconstruction tool, suitable also for non-expert users, developed in the ambit of the EPOCH Network of Excellence and quite diffuse between the Cultural Heritage community. The second is a very interesting tool, similar to the Arc3D but that require a bit of manual intervention and expertise to produce a complete model starting from a set of image. In practice, VisualSFM is a GUI that assemble in a very efficient way several tools released under GPL license for the image-based reconstruction and allow the user to inspect the different phases of the models reconstruction. Both the output format of the Arc3D and the VisualSFM tool can be imported for further elaboration into Meshlab.

5.1.2 3D viewer for digital asset preview

A generic 3D viewer is required to provide a preview for the visualization of the results of searching operations in the V-MUST repository. This viewer should be web-based, hence based on X3DOM and/or SpiderGL and

capable to visualize colored models (both textured and with color per vertex attribute). Since its main use is intended for previewing 3D data the overall performance can be limited and a simplified version of the model can be used, also to reduce data transmission. Just to give an idea about how this would looks like, the project ADL 3D Repository (http://3dr.adlnet.gov) provides an interesting WebGL-based interface for browsing public 3D content (see Figure 5.1). We can take inspiration from this (or similar) project for the initial design of this preview tool.

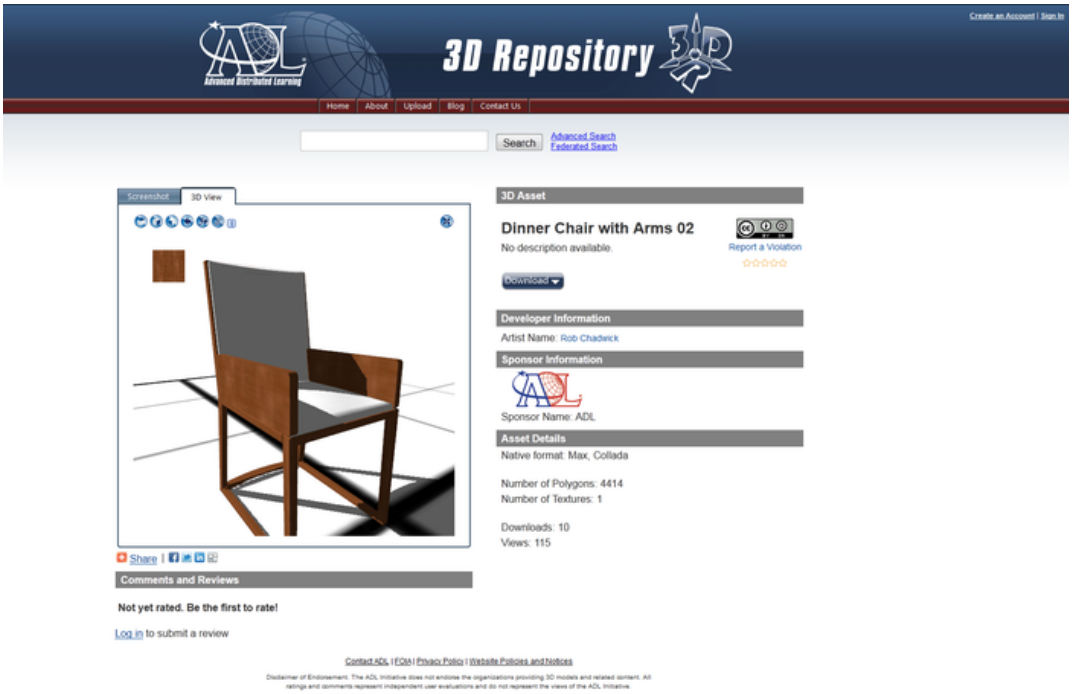


Figure 5.1 A 3D preview interface for searching of 3D content from the ADL 3D Repository project.

5.1.3 3D viewer for remote collaboration

Another 3D viewer required is the one to support remote collaboration, for example some working team's element may want to present 3D data (or images, or video) to other working team's elements during the development of a certain project. This viewer should be capable to display different multimedia data and, concerning 3D data, complex scene and hence it has different requirements with respect to the one just

mentioned. The technology behind this viewer will be most probably X3DOM or something similar to OSG4WEB since potentially complex scene should be visualized. Additionally, this viewer will constitute the base to develop other viewers for other tools, like the pre-visualization engine of the storyboard editing tool.

5.1.4 Storyboard editing tool

This feature is ambitious but it is required by a lot of partners as a part of the Virtual Museum production to enable the storytelling creation step to a collaborative, accessible, easy updatable tool and starting point for every other content. One of the main objective is, at the minimum, to provide a pre-visualization tools like the one typically employed in storyboard creation to assist the creation of the story in an user-friendly manner. The interface of this tool will be designed taken into account what stated in the section about the user interface design to be really effective. This pre-visualization engine shares similar functionalities with the just mentioned viewer for remote collaboration and hence it can be derived from it or vice-versa.

5.1.5 Scene editor to assemble assets in virtual environments

An important operation for prototyping, for collaborative hypothesis discussion is the possibility to quick assemble 3D objects to compose a virtual environment. The possibility to assemble different digital assets should not limited to 3D models only but also to other digital assets (e.g. images, video) in order to enrich the final virtual environments. This is a fundamental feature that should be used also by the storyboard tool to delineate well the pre-visualization phase. This scene editor is not necessarily an online editor but in this last case this tool can become a valuable content editor also for any x3DOM-based graphics application, not limited to Virtual Museum projects. Also in this case the user interface plays a very important role to make the tool itself really effective.

5.1.6 A social exchange service while reconstructing for the production phase with advanced visualization capabilities

A dedicated social exchange service is required by some partners to improve the reconstructing step and, generally, all the reviews of the assets during the workflow. The possibility to review step by step the interpretation and the reconstruction hypothesis by authorized external members of the team (i.e. consultants, etc.) would be a valuable feature. Some examples of open source platforms that make this possible are *elgg* (http://elgg.org) and *pligg* (http://pligg.com); closed solutions like *jive* (http://www.jivesoftware.com) could be a reference also. In this context could be very useful to integrate this "social exchange" service with a 3D viewer

to make possible to "post" a reconstruction hypothesis (or also a single asset) to discuss about it and finding a collaborative validation. Despite other tools/components this is more a valuable feature here discussed than a concrete planned one.

5.1.7 Virtual Museum Evaluation Wizard online tool

This feature matches the truly last step of the Virtual Museum production. A Virtual Museum evaluation online tool could be very useful for helping the team who create the project to have a clear feedback from the application ready to use in the next Virtual Museum's concept step. This tool will be developed according to the output of the WP7, that is specifically involved in the development of guidelines for the evaluation of Virtual Museum. At the present moment, the best choice seems to base this *Evaluation Wizard* on a template basis to account for the different Virtual Museum domains.

5.2 Other tools and components

The tools and components described in this section does not come from the analysis of the V-MUST technical questionnaire or from the V-MUST partners'' workflows but are valuable tools that worthwhile the inclusion in the CIF, especially the expressive rendering techniques usually very useful for the visual inspection of the surface' details of Cultural Heritage artifacts.

5.2.1 Community Presenter

The Community Presenter is a visualization tool developed in the ambit of the 3D-COFORM European Project (FP7) for the visual presentation of collections/exhibitions.

Current main features are the following:

- easy setup of media visualizers:
  - 3DS mesh models
  - multiresolution nexus files
  - multiresolution images

- seamless integration with web technology
  - add a viewer to an existing website,

- load webpages inside viewers
- streaming remote content

- templates for common cases
  - single model viewer
  - small collection of objects

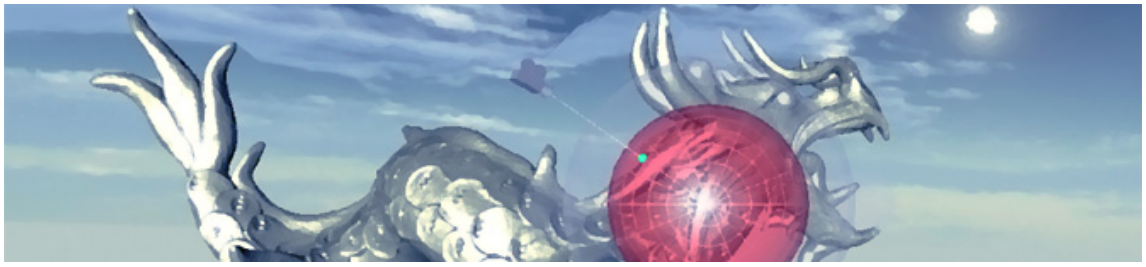
The technology behind this tool (it is heavily based on QML) can be easily adapted for the needs of the V-MUST platform, in particular for the quick presentation (with different styles) of 3D object collections. An example of application is shown in figure 5.2, where the Community Browser is used to present the collection of images and the 3D models of capitals acquired in the ambit of the Cenobium project (described in Section 2.7.1).

Figure 5.2 An example of application of the Community Browser to show the capitals' collection of images and 3D models of the CENOBIUM project.

5.2.2 Interaction Tools - Navidget and tBox

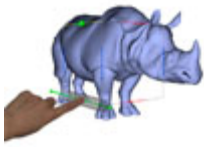
All the following tools only required a 3D object and a 3D rendering engine. They are perfectly compatible with the common tools in use in V-MUST.net consortium and will be developed as libraries/plugins that can be re-used in different contexts.

- **Navidget** (http://dx.doi.org/10.1145/1450579.1450589)



NavidgeIt is a new interaction technique for camera positioning in 3D environments. Unlike the existing POI techniques, Navidget does not attempt to automatically estimate where and how the user wants to move. Instead, it provides good feedback and control for fast and easy interactive camera positioning. Navidget can also be useful for distant inspection when used with a preview window. This new 3D User interface is totally based on 2D inputs. As a result, it is appropriate for a wide variety of visualization systems, from small handheld devices to large interactive displays. A user study on TabletPC shows that the usability of Navidget is very good for both expert and novice users. Apart from these tasks, the Navidget approach can be useful for further purposes such as collaborative work and animation.

- **tBox** (http://dx.doi.org/10.1145/1978942.1979387)

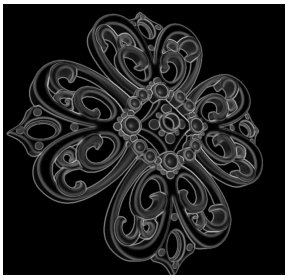


3D transformation widgets are commonly used in many 3D applications operated from mice and keyboards. These user interfaces allow independent control of translations, rotations, and scaling for manipulation of 3D objects. In this paper, we study how these widgets can be adapted to the tactile paradigm. We have explored an approach where users apply rotations by means of physically plausible gestures, and we have extended successful 2D tactile principles to the context of 3D interaction. These investigations led to the design of a new 3D transformation widget, tBox, that can been operated easily and efficiently from gestures on touch-screens.

5.2.3 Visualization tools for expressive rendering

All the following tools only required a 3D object and a 3D rendering engine. They are perfectly compatible with the common tools in use in V-MUST.net consortium and will be developed as libraries/plugins that can be re-used in different contexts.

- **Apparent Relief (**http://dx.doi.org/10.1145/1377980.1377987)

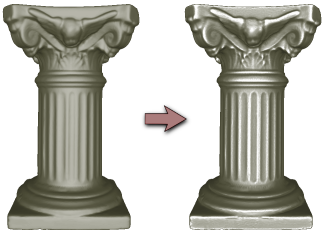


Shape depiction in non-photorealistic rendering of 3D objects has mainly been concerned with the extraction of contour lines, which are generally detected by tracking the discontinuities of a given set of shape features varying on the surface and/or the picture plane. In this paper, we investigate another approach: the depiction of shape through shading. This technique is often used in scientific illustration, comics, cartoon animation and various other artwork. Our solution is a novel view-dependent shape descriptor called Apparent Relief, which carries pertinent continuous shape cues for every pixel of an image. It consists of a combination of object- and image-space attributes. Such an approach provides appealing properties: it is simple to manipulate by a user, may be applied to a vast range of styles, and naturally brings levels-of-detail functionalities. It is also simple to implement, and works in real-time on modern graphics hardware.

- **Light Warping** (http://dx.doi.org/10.1145/1531326.1531331)

Recent research on the human visual system shows that our perception of object shape relies in part on compression and stretching of the reflected lighting environment onto its surface. Our tool uses this property to enhance the shape depiction of 3D objects by locally warping the environment lighting around main surface features. Contrary to other tools, which require specific illumination, material characteristics and/or stylization choices, our approach enhances surface shape without impairing the desired appearance. Thanks to our novel local shape descriptor, salient surface features are explicitly extracted in a view-dependent fashion at various scales without the need of any pre-process. The warping itself is very fast to compute on modern graphics hardware, enabling real-time performance in direct illumination scenarios.

- **Radiance Scaling** (http://dx.doi.org/10.1109/TVCG.2010.252)



Based on the observation that shading conveys shape information through intensity gradients, Radiance Scaling modifies the classical shading equations to offer versatile shape depiction functionalities. It works by scaling reflected light intensities depending on both surface curvature and material characteristics. As a result, diffuse shading or

highlight variations become correlated to surface feature variations, enhancing concavities and convexities. The first advantage of such an approach is that it produces satisfying results with any kind of material for direct and global illumination: we demonstrate results obtained with Phong and Ashikmin-Shirley BRDFs, Cartoon shading, sub-Lambertian materials, perfectly reflective or refractive objects. Another advantage is that there is no restriction to the choice of lighting environment: it works with a single light, area lights, and inter-reflections. Third, it may be adapted to enhance surface shape through the use of precomputed radiance data such as Ambient Occlusion, Prefiltered Environment Maps or Lit Spheres. Finally, our approach works in real-time on modern graphics hardware making it suitable for any interactive 3D visualization.

5.3 Services

In this section we briefly describe some services that can support the needs/requirements previously mentioned, like a service to ingest 3D data into the digital assets repository, a remote render farm service that can be employed also by the storyboard editing tools to improve the visual presentation of the pre-visualization engine, and a visual interface for metadata ingestion that can help the users to quickly and efficiently provide a description of his/her digital assets during the workflow (or in the archiving phase) through a suitable definition of metadata.

5.3.1 Automatic deploy of 3D content

The automatic deploy of 3D content is a complex task to achieve from a technical viewpoint. With this service we intend a web service that allow the user to upload his/her digital 3D model into a repository of digital assets. The main complexity behind the realization of a similar service is that different 3D data formats have different limitations and description capabilities that influence heavily the overall architecture, in terms of 3D previewing, the re-use and the search operations. For example some format can support color-per-vertex while other formats cannot, some can include camera description parameters to allow the object to be re-used in a shot of a 3D scene while other does not include such information, and so on.

In particular if we want to provide a 3D preview (as stated in section 5.1.4) of the content of the repository the service should analyze the 3D object uploaded and inform the user about the potential problems found, for example topological anomalies. Meshlab just provides similar functionalities both to analyze the model identifying topological problems and to remove such problems in an automatic way. Obviously, problems related to the specific data format are more difficult to solve. These are important aspects if we want to

guarantee that other users can visualize correctly the model archived. In order to preserve the original information provided the 3D object uploaded can be stored in two different version in the repository; the original one plus a converted version more suitable for the preview. Additionally, during the parsing of the original data some useful information about the model can be extract and stored as metadata to make subsequent search operation more easy and effective.

5.3.2 Visual Interface for metadata ingestion

For each digital assets, like as images, video, documents, 3D objects, and so on, it is required to associate some information to them in order to support search and retrieval and other processing operations. Some of this information can be extracted automatically, like the resolution of an image for example, while others, like metadata regarding IPR of the digital asset, must be provided by the users during the ingestion phase. Since, many times a complete description in terms of metadata could be very complex, like if the user want provide digital provenance of the asset to permit subsequent semantic reasoning on it, it is necessary to design an instrument to facilitate the user in providing the information required. This visual interface should support the archiving phase of the assets.

5.3.3 Remote render farm

In order to support some workflow that deals with extensive movie rendering of complex assets we plan the experimental deployment of a remote rendering service. Initially, this services will be provided to interested partners on a limited resource base, hosted on the HPC Cineca cluster. In the first implementation, Blender will be the rendering platform to build such service, as it is open source and widely used. In the first implementation the service will be deployed as an additional service available in the central production infrastructure.

5.4 CIF Integration Mechanism

Here, we provide some indications from a technological point of view about how the different tools/components and services will be integrated within the CIF. One of the first aim is to support sustainability and re-usability of the tools. In this direction, we opt as most as possible for open source tools and open data formats to reach this goal.

In the following we present some of the integration mechanisms the CIF will employ to provide tools/components/services interoperability. In particular, first we describe how interoperability between different tools will be achieved through data format conversions, i.e. *transcoding*, then we discuss something about how to integrate different visualization technologies for the development of powerful and flexible

graphics applications. At the end we provide a simple example of integration through a declarative presentation workflow based on X3DOM.

5.4.1 Transcoding

As previously stated, one of the main component of the CIF will be a set of tools for the automatic (or not depending on the cases) conversion between different data format in order to increase the interoperability of the tools/components of the CIF. In order words the conversion between different data format is the ``glue'' to connect different tools/components, allowing inter-operability between them. As described also in the service for the automatic deployment of 3D content (Section 5.3.1) conversion will play an important role to connect also visualization tools and the digital assets repository.

We recall here that from this and the previous analysis the data format supported will be :

- 3D Studio Max data format (3DS)
- Blender data format
- COLLADA
- PLY data format
- X3D data format
- Bundler output data format
- V3D data format
- Common image data format: JPEG, TIFF, PNG, ..

The 3DS, Blender files, COLLADA and PLY format plays an important role in the integration of 3D processing and modeling applications/services.

The V3D and Bundler data format are required to support the archiving and processing of image-based reconstruction tools such as the ones previously described, i.e. the Arc3D and the the VisualSFM tools.
The conversion between the different image format will be achieved mainly through scripts (for example written in Python) and common conversion tool such as the ImageMagick one (http://www.imagemagick.org/script/index.php).

5.4.2 Visual technologies integration

In chapter 4 we describe some visualization technologies that who are interested in for the development of the CIF. The different visual technologies can be integrated in an elegant way to maximize their potentials. We

think at two types of integration, one for the stand-alone tools and another one for the web-based applications and services.

For **stand-alone applications**, the integration consists in the development of libraries (to ensure re-usability outside the CIF) and plugins for the software used in CIF (like the MeshLab tool). Such a library exists for Navidget and is currently updated to latest standards of developments. A plugin for MeshLab for Radiance Scaling is just available and will be presented at the next CAA 2012 conference. We need to point out that some tools, dedicated to some special tasks and endpoint of the pipeline such as ArcheoTUI (http://dx.doi.org/10.1145/1512714.1512761) may only be integrated on the CIF by relying on its technology only.

For **on-line solutions**, javascript libraries is the main selected solution to integrated these technologies into the CIF. By relying on such a standard, we ensure the re-usability of the results issued from V-MUST.net project. Indeed, a webGL/SpiderGL library for the Navidget software interaction tool will be ready in the upcoming months. For what concern visualization, special care must be taken. For example, for pure image-based techniques such as the Apparent Relief, Light Warping, and Radiance Scaling, an implementation relying on SpiderGL is the recommended solution. For more complex 3D scene X3DOM is the preferred solution. In this case, for example to include expressive rendering modes, new node specification into the graph-scene standard will be defined. The same is valid also for other advanced visualization features, for example huge mesh rendering can be achieved into a complex scene by providing a new node specifically designed (in WebGL/SpiderGL) to deal with this purpose into an X3DOM.

5.4.3 Declarative Content Presentation Workflows based on X3DOM

Besides presentation, i.e. the rendering and user interface part, workflow issues must be considered, too, including tools and tool-chains as well as content and media authoring. While declarative representations help reducing the application development and maintenance efforts, the content first needs to be generated somehow. In general X3DOM is extremely helpful for application- or domain-specific production pipelines. Defining the common 3D geometrical format as X3D the CIF mainly relies on an open ISO standard that is a superset of the older VRML ISO standard and which is supported by a large and growing number of Digital Content Creation (DCC) tools. It comes with a formal description of content but also a definition on an abstraction of the underlying run time environment enabling the realization of customized applications. Second, the X3DOM project itself provides a bundle of online- and offline-tools (e.g. plugins and re-coder, see Figure 5.3) to ease the production and processing of content items. Besides all these techniques the project provides

also software components, tutorials, and examples on the web page, which explore and explain how to get the data from a specific DCC Tool, e.g. Maya or 3ds Max (Autodesk, 2011), into your 3D web application.

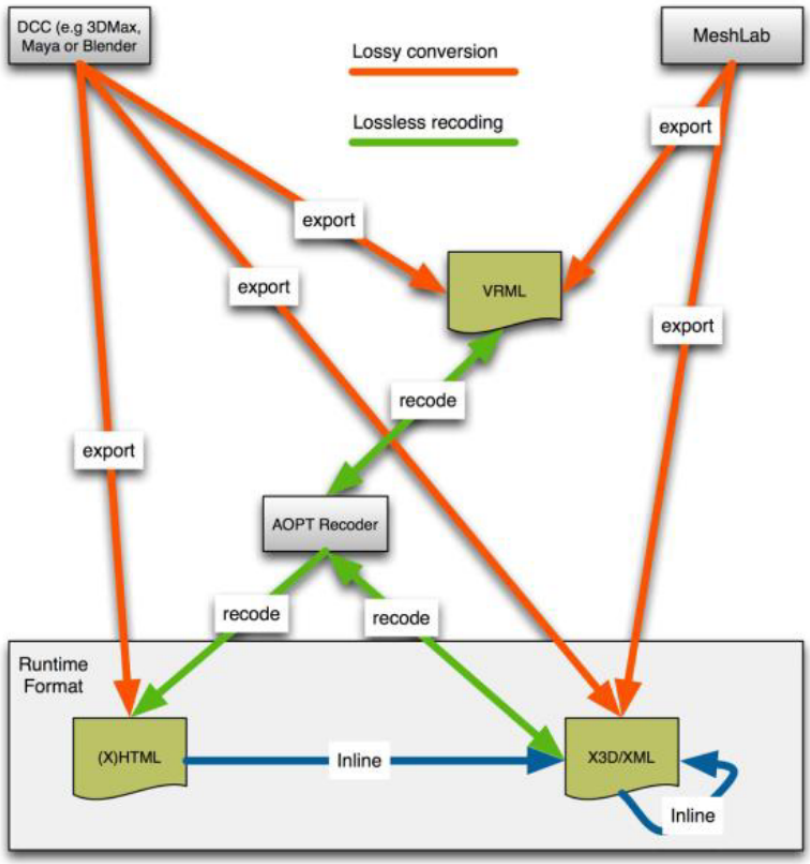


Figure 5.3 Interactive tools to export and recode data for X3DOM, e.g. using MeshLab, as one major VH tool.

In virtual heritage, MeshLab (http://meshlab.sourceforge.net/) is an important tool to process and manipulate mesh datasets, which in addition can already export the 3D data into the X3D format, including textures, vertex colors, etc. However, when dealing with 3D scans the vast amount of data is an issue for several reasons. WebGL only supports 64k indices per mesh and therefore large models have to be split. X3DOM splits this automatically if necessary, but besides the memory footprint, loading the data, especially over the web, still takes time. Hence, data reduction should be considered as well. While progressive meshes and similar level-of-

detail techniques are applicable here, the original set of normals and colors of the high-res mesh must be preserved for appropriate visual quality, wherefore normal and color maps can be used.
Another issue in the content pipeline one need to think of is annotations and metadata processing. A possible scenario here is 3D content that shall be annotated with metadata to allow for interlinking and concatenation with further information and additional content like HTML sites, multimedia, etc.

5.5 CIF Architecture for the Visualization and Interaction on the Web

Within this section, we focus onto automatable processes for web application development by designing and implementing a fully automated Web Service Portal, which provides web services that automatically combines application templates and raw data into interactive 3D visualizations for the Web. This is in line with the endeavors in WP4 offering an integrated service platform embedding the CIF. As mentioned above the CIF for standalone applications will be a harmonized pool and collection of tools enabling data transfer and semi-automatic workflows for the creation of Virtual Museum assets. In view of the deployment we refer the reader to deliverable 4.2.

5.5.1 Web Services and Transcoding

The goal is to have web services that automatically convert raw 3D data (e.g. point clouds, meshes, and architectural/archaeology models) into interactive 3D visualizations for the Web that can be delivered as a cloud service. This includes object visualization, metadata presentation as well as the provision of adaptive GUI elements for object exploration or camera based navigation. This enables the user (in our case Virtual Museum actors, see deliverable D4.2) to interactively explore the presented 3D data. The user must be able to directly interact with the visualization, e.g. by selecting a certain region for which he wants to obtain more information (e.g., show buildings, objects and trees in a marked area) or by clicking onto a certain POI, which in turn delivers other information from the processing backend or associated metadata. Therefore, the visualization needs to be scalable in such a way that mobile and desktop machines with rather different computing and 3D capabilities are supported likewise, for example via an hybrid approach that provides both, streaming for low-end clients and direct web-based 3D rendering for high-end machines. Moreover, cross-platform and cross-browser issues must be addressed. Finally, security aspects related to IPR protection are of high importance. Here again, streaming technologies can help with information hiding in that only image streams but no 3D models are transferred over the network to the client for display (see deliverable 2.5 for an in depth discussion about this topic).
As mentioned above, the proposed architecture will be integrated into the service platform defined in Deliverable 4.2. In order to scale the solution web services are envisaged to enable coherent transmission and

communication schemes with the V-MUST platform. These also allows using the services in broader software ecosystems, which automatically exchange data. Moreover, declarative, XML-based languages such as X3D are suited well for SOAs in that 3D content (such as spatio-temporal data or a given WCS response in a geo-spatial application) can be directly transformed, e.g. via an XSLT or similar transform, from one representation to another (such as an X3D world that can be rendered in real-time in the web browser using X3DOM). This transformation is done with the help of a transcoder service, which also needs to consider the respective client capabilities (e.g. mobile device vs. desktop system) to provide an appropriate visualization. For being able to present even large-scale big data, suitable compression schemes are required for both, the client (especially when considering mobile devices) and the server (where latency is the main issue). In our implementation this is handled by utilizing image-based compression algorithms that can encode geometric as well as material data. In addition, appropriate caching strategies allow for faster delivery of 3D contents.

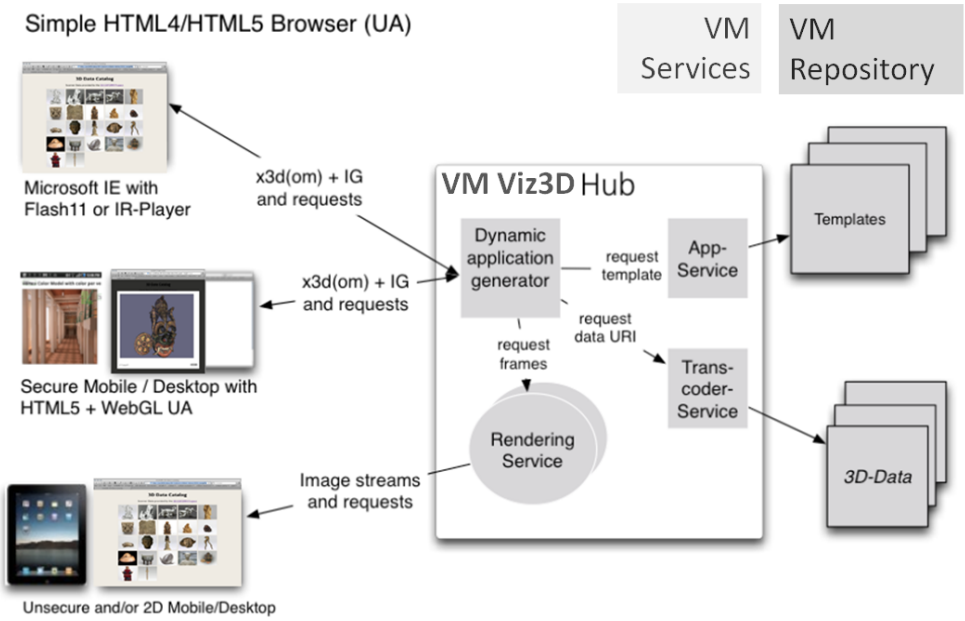


Figure 5.4 Architectural Layout of CIF embedded as Web Services and deployed in the V-MUST platform

Figure 5.4 shows the concept of the planned service oriented architecture. The core component is the "Virtual Museum Viz 3D Hub" that acts as application provider, whereas the application identification is resolved via an

URI. The "app" service takes a request from the hub and extracts an application template, which species the data and application characteristics. For data preparation, the template provides at least the HTML(5) pages, the necessary metadata of the respective Virtual Museum application, as well as descriptions of all required data containers, e.g. in X3D format. Thus, the Hub is able to invoke the appropriate transcoder service for providing the concrete data (e.g. a 3D world with certain camera positions). The transcoder service translates and, where required, caches the 3D data from already existing descriptions by converting them from various formats, which are not yet suitable for real-time web presentation, to a declarative deployment format for final presentation. Here we use X3D in combination with HTML5, since declarative formats nicely scale to more general SOA pipelines. The result is then provided as URI.

5.5.2 Client-side Rendering

Depending on data security aspects and the respective web browser capabilities, the Hub generates a concrete Virtual Museum application for presentation and interaction on the web. One possible solution is based on client-side rendering by utilizing e.g. X3DOM, which either uses JavaScript with WebGL or Flash 11 with Stage 3D for real-time rendering. Open research issues here are scalable methods for binary compression of big geometric data (e.g. by utilizing the proposed image geometry approach) and of material data, as well as suitable caching strategies. The advantages of client-side rendering are a very simple server infrastructure and highly interactive applications since everything is rendered on the client. Disadvantages are that the data-load can easily overburden the client, that large 3D data sets need to be transferred, to comply with the security or IPR issues. Novel streaming solutions should be developed to account for this disadvantage.

5.5.3 Server-side Rendering

A second option is server-side rendering. In this case, for instance an X3D runtime environment is utilized for rendering and the rendered image frames are transferred (e.g. as MJPEG stream) to the client. Since interactions are handled via WebSockets or XMLHttpRequest, the latency is much higher than for client-side rendering. Therefore, we are currently exploring suitable interaction methods and message protocols. Another disadvantage is the fact that a complex server infrastructure is required. However, since no 3D data but only images are transferred over the network, there are no IPR issues concerning the 3D data and data security is inherently given as a nice side effect. Also, the visualization application can be executed on arbitrary clients, which is another advantage of server-side rendering.

Some additional details about 3D model security and rendering for both the client-side and the server-side approach can be found also in Deliverable 2.5.

6 Portability

In this section we discuss some issues concerning portability. In the first part, we discuss about the fact that the CIF is designed to guarantee portability not only of the content that is capable to handle, but also of the visualization tools themselves that can be executed on many different platforms. This is due to the technologies used for their implementation. In the second part we discuss some issues about touring installations and the portability of displays, presenting first a tool for setup different configuration of projection-based display and then providing some recommendations about this topic.

6.1 Standards

The CIF is designed to ensure maximal portability and re-usability of the components/tools both from a technological point of view and from an user point of view. Using standard formats being able to scale to different platforms the CIF allows to port implementation across hardware platforms.
One of the standards we follow is to use WebGL-based implementations for the various web visualization solutions making applications on the Web able to work on very different devices and systems. The WebGL standard is designed to reduce hardware requirements because it can be implementable on almost any modern devices with standard graphics capabilities.
In view of the interportability of the tools and components provided by the CIF, we often stated during this document that many of the tools/components/services rely on standards (like X3D) or open data formats (like COLLADA or the Blender data format). The adoption of standards and open data formats guarantee, that the output of many of the tools/components of the CIF can be reusable in other upstream or downstream tools/applications. Moreover, this ensures a long lifecycle for the CIF. This fact also results due to the design choice of open source implementations as much as possible ensuring wide-diffuse open source tools/components that have greater possibility to be sustained during the next 5-10 years.

6.1.1 X3D

The X3D ISO Standard does not only provide a file syntax and different encodings (e.g. classic, XML and binary) but also semantics which define how the scene must be rendered and shaded and how the scene should be executed according to a well specified runtime and event model (http://www.web3d.org/files/specifications/19775-1/V3.2/index.html).

Conceptually, the semantics of X3D describe an abstract functional behavior of time-based, interactive 3D, multimedia information and do not at all specify a specific software or hardware setup.

X3D features a core component of the conceptual part of ISO/IEC 19775. The Core component supplies the base functionality for the X3D run-time system, including the abstract base node type, field types, the event model, and routing mechanism. One of the main concepts of X3D is the sensor concept enabling events to be triggered and processed. Here, the X3D specification includes only high level sensors. The X3D sensors recognize, that "the user" has touched or turned some part of the world but do not specify how a specific input device or user action has to be handled. The X3D specification of high level sensors has to be reinterpreted for immersive environments. However, it supports this profile allowing X3D being used in several large scale installations. Additionally, the system also provides low level sensors that stream data (e.g. stream of float values) from the real to the virtual world and vice versa enabling a scaling to smaller mobile devices. Here the standard provides several features:

- The system utilizes the concept of nodes and routes not only for the scene itself but also for the browser configuration and any other dynamic aspect. Every dynamic component (e.g. Job, Window) can be implemented as a node with fields, communicating with slots living in a specific namespace and hierarchy. The namespace type defines the use and the types of nodes that can live in the namespace. The route mechanism not only defines the communication channels directly, but also the thread paths used to parallelize the system execution indirectly.
- The geometrical standard allows the developer to create prototypes with behavior scripting in Java or JavaScript. In order to achieve maximum performance, we need the ability to extend the node pool with native C++ implementations and fetch these nodes without recompiling or linking the toolkit.

Thus X3D provides adequate mechanism to establish scalable interactive graphics systems.

6.1.2 Abstract X3D Structure and Organization

An X3D world is conceptually defined as a sequence of statements organized conceptually as a file. The first item in the file is the Header statement. The second item in the file is the PROFILE statement. The PROFILE statement may be optionally followed by one or more COMPONENT statements. The remainder of the file consists of the other elements defined in ISO/IEC 19775. Optional META statements follow any COMPONENT statements or, if there are no COMPONENT statements, the PROFILE statement. All other statements follow the statements described above. ROUTE statements are used to specify the pathways for allowed transmission of events. These statements link a field in one node to a field of the same field type in another node.

PROTO statements are used to specify new node types. Such statements assign a name to the new node type along with a declaration of the interface for the new node type. This is followed by a definition for the node type functionality. EXTERNPROTO statements are used to specify an interface to PROTO or EXTERNPROTO statements located externally to the local file. Any additional X3D content loaded into the scene via Inline nodes or scenes loaded using createX3DFromStream, createX3DFromString, or createX3DFromUrl, shall be declared as having a profile that has an equal or smaller set of required functionality; i.e., there can be no components explicitly declared, or implied by the profile in that content, that requires functionality not declared in the original profile and component declarations for the containing scene.

6.1.3 Description of RTE

The X3D run-time environment maintains the current state of the scene graph, renders the scene as needed, receives input from a variety of sources (Sensors) and performs changes to the scene graph in response to instructions from the behavioral system. The X3D run-time environment manages the life cycle of objects, including built-in and user-defined objects and programmatic scripts. The run-time environment also manages interoperation between the X3D browser and host application for file delivery, hyperlinking, page integration and external programmatic access.

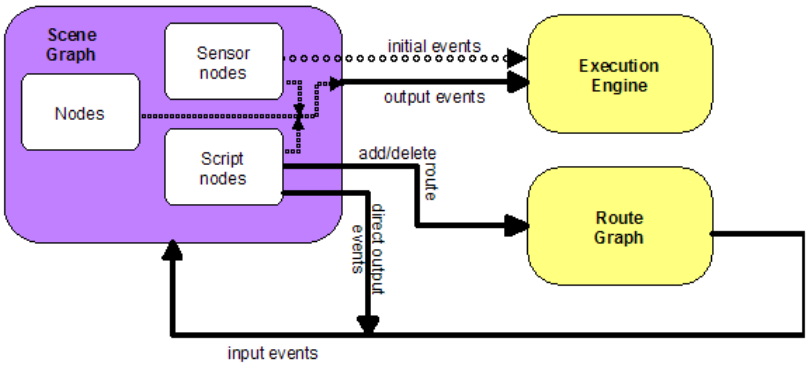


Figure 6.1 Conceptual RTE Design of X3D

The X3D run-time architecture is independent of the data encoding format. X3D content and applications can be authored in a variety of encodings, including textual (XML and Classic VRML encodings) and binary, either compressed or uncompressed. ISO/IEC 19775 contains an abstract encoding specification that defines the

structure of the X3D scene: hierarchical relationships among objects, initial values for objects, and dataflow connections between objects. All concrete data encodings for X3D conform to this abstract specification.
Browsers and generators may support any or all of the standard encoding formats, depending on their application needs and the conformance requirements of a specific component or profile.
X3D encodings are fully specified in the parts of ISO/IEC 19776.
The main advantage of the RTE is that an author of X3D content can control the creation and management of scenes, rendering and behavior, and loading of media assets. The loading and incorporation of authored extensions, which can be written in X3D or an external language, can also be controlled. The ability to make content-defined extensions is provided in profiles that support prototyping mechanism.

6.1.4 Profiling ensuring Interoperability and Scalability

As mentioned in section 6.1.1, X3D supports the concept of profiles. A profile is a named collection of functionality and requirements that shall be supported in order for an implementation to conform to that profile. Profiles are defined as a set of components and levels of each component as well as the minimum support criteria for all of the objects contained within that set.

ISO/IEC 19775 defines seven profiles satisfying varying sets of requirements:

- *Core profile* ensuring absolute minimal file definitions required by X3D
- *Interchange profile* enabling the exchange of geometry and animations between authoring systems as well as addressing the limitations of software renders not capable of dealing with all details of the full X3D lighting model
- I*nteractive profile* implementing a lightweight playback engine that supports rich graphics and interactivity and a low-footprint engine requiring limited navigation and environmental sensor control (e.g. an applet or small browser plug-in),
- *MPEG-4 interactive profile* providing the base point of interoperability with the MPEG-4 standard
- *Immersive profile* implementing immersive virtual worlds with complete navigational and environmental sensor control
- *CAD Interchange profile* distilling computer-aided design (CAD) data to downstream applications
- *Full profile* ensuring X3D fully implemented

In order to allow X3D being portable onto mobile devices, X3D introduced the concept of namespaces which provide access to the key functionalities of render and hardware support. Thus, it allows developer to

implement a *Mobile Profile* integrating different namespaces being able to reflect on the underlying hardware and render restrictions.

6.1.4 X3D, Portability and Deployment

All these specifications are independent of any CPU or GPU infrastructure or API. The content is portable across different runtime environments (e.g. Windows, Mac), platforms (e.g. mobile, desktop) and render architectures (e.g. OpenGL or Raytracing). This is a big advantage compared to related deployment models like commercial plugins (e.g. Flash, very limited support on Mobile devices), content-bound-plugins (e.g. Unity3D, works only on the specific browser/CPU infrastructure) or Browser API's (e.g. WebGL (bound to OpenGL ES 2.0) which can not be mapped e.g to a raytracing infrastructure).

6.2 Display Configuration for Touring Installations

6.2.1 Projection-based displays

Unlike traditional displays, projection-based screens can have different aspect-ratios and sizes. Moreover, it can take different forms in space (cylindrical, partially spherical, spherical, free form) which makes it the perfect solution for immersive systems.
Immersive displays, besides other components in Virtual Museums such as interaction methods, are means of achieving the feel of being there (presence) in a museum or heritage site. The more the display is immersive the more the Virtual Museum user is convinced of his "virtual" presence experience. Taking into consideration that the human eye's field of view FOV is about 180°, we will be able to understand why panoramic displays and CAVE's increase the feel of immersion.

Culturama 2 as a case-study

Culturama2 is a projection-based display developed at CULTNAT. It uses 3 HD projectors to fully cover the half-circular display area. To do so, each projector is required to project a pre-distorted image on a 3 adjacent screens.

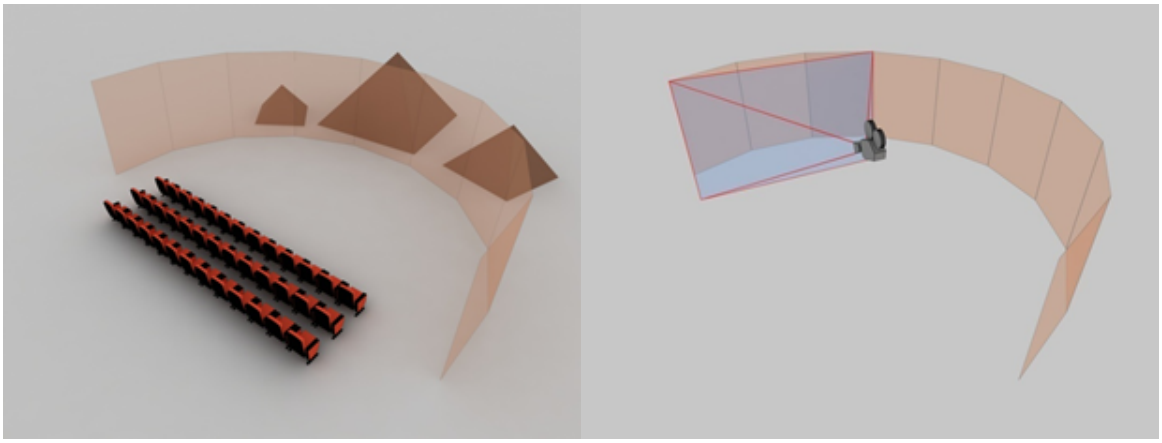


Figure 6.2 (Right) Culturama2. (Left) Projector positioning.

Displayed content

Content that can be viewed through projection-based displays are generally divided into two types. The 3d environment that represents reality and other non 3d data that can be 2d information, menus, cursor … For the second type, the information is always represented in the screen-space (parallel to the display screen)

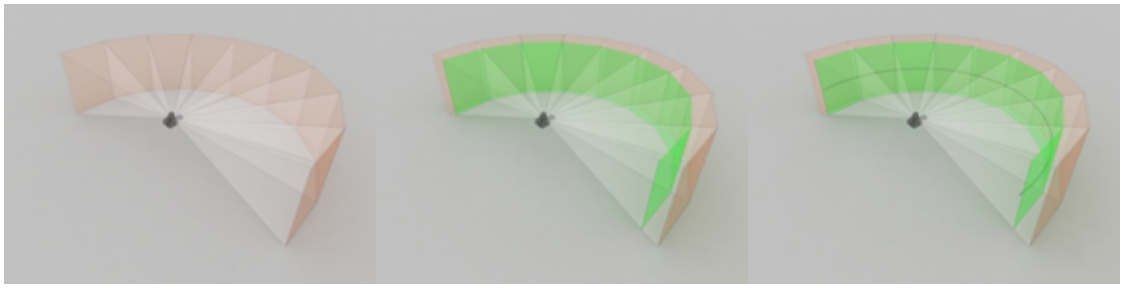


Figure 6.3 (Left) Panoramic display. (Center) Screen space. (Left) Track mouse pointer.

Display configuration

Culturama is a customizable system. It can be adjusted to fit different FOVs by changing the angles between screens. It can also be customized by controlling the number of screens that compose the display (3, 5, 7…). To make sure the content, 3d environment and 2d screen-space, are not affected when re-configuring the display, a special component has been developed that acts as an interface to receive the required display configuration

(width, height, angle between screens…) and automatically adjust other component in the scene, like for example the cameras that sees the 3d environment.

6.2.2 *3SIV* as a case-study

In the Virtual Reality Lab at Lund University, we have completed the construction of a visualization set-up with the working title *3SIV* (Triple-Screen Immersive Visualization). 3SIV was built as a simple, low-cost supplement to our CAVE system (which is expensive, delicate, and requires that four computers work in a cluster).
Using a single computer, 3SIV is easier to adapt with existing software for visualization, simulation and game development. It is also meant to be easier for researchers, teachers and students, to first develop and test run simulations on their own computers, and then after a simple modification, directly be able to run their simulations in 3SIV.
The advantage of three large screens is to achieve an immersive experience that covers the entire field-of-view. In addition, active stereoscopy can be used to further enhance the experience. Unfortunately, there is not yet a generally accepted standard for active stereoscopy, so we've had some trouble getting the video drivers (NVidia's 3D vision) to accept our 120Hz-projectors. The problem was solved with writing a faked driver file for a generic CRT-screen. However, there is still an issue with the lack of a "stereo projection eye-swap" in NVidia's software. We still hope, though, that we can find a workaround to this, or that NVidia decides to support such a feature in future drivers.

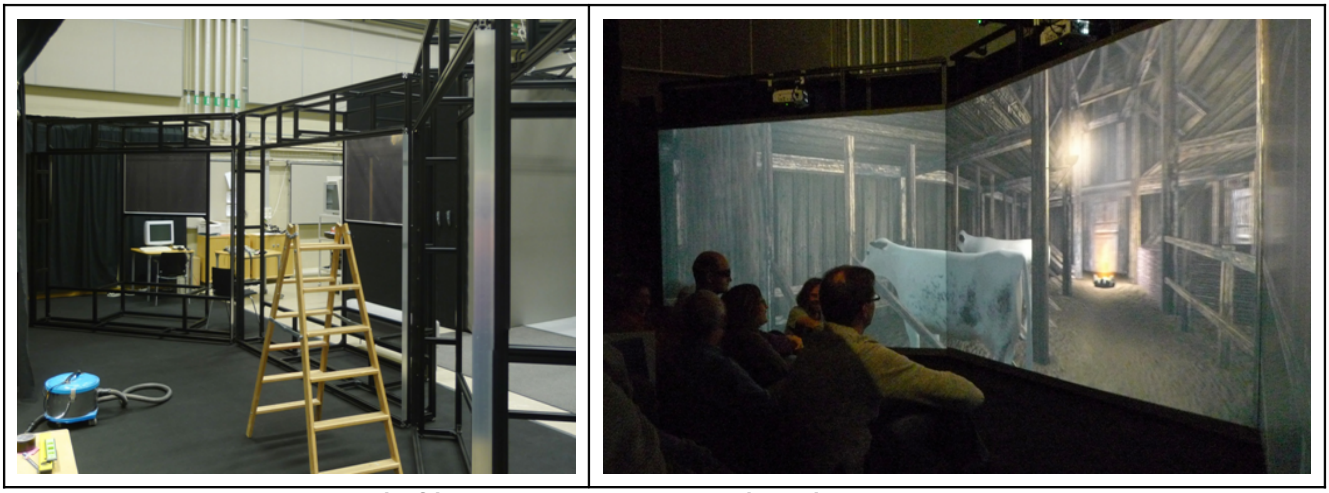


Figure 6.4 (Left) 3SIV under construction (Right) 3SIV in action.

6.2.3 Recommendations

The main expectations for a portability requirement are to:

- Ensure that the components and the overall infrastructure can be flexible, reusable and easily and quickly ported to specified new environments if necessary.
- Minimize porting costs and schedules

The following guidelines have been found to be useful when producing portability requirements:

- Because the amount of effort required porting an application or component typically depends on the Environment and Technologies to which it is being ported, portability requirements should specify the potential environments to which the application or component will be ported. Of course physical aspects of the installation cannot be discuss at this point, but regarding Technology issues, working with standards it is a key question to avoid portability problems between technologies..
- Avoid redundancy between portability and interoperability requirements.
- Porting requirements though standards should be necessarily specified in terms of architectural, design, and implementation constraints and the use of industry best practices.

7 Micro Projects

The DoW of V-MUST includes a call for micro-projects, to be developed in the framework of WP4 and WP5. Such micro-projects will be proposed by the partners and selected according to criteria which maximize their utilities in the context of the development of the V-MUST platform.

This call was initially planned for the end of February 2012, but it has been postponed to the end of April 2012 in line with the call for testbeds (WP7), in order to ensure the coherence of the setup envisaged in WP7 and the early implementations proposed within the scope of the presented CIF.

The goals that the micro-projects must fulfill have to guarantee congruence with the proposed CIF tools but also should be built on common grounds with WP7 interactive lab set-ups exposing different demonstrators. This will enable us to finalize a more effective selection among the micro-projects proposal.

Despite this, we will focus in the upcoming months on the further development of the enabling building blocks, in order to allow the CIF being fully functional. A first alpha version wrapping advanced functionalities is foreseen for month 18. In particular, we aim to show an expressive rendering techniques integrated in the X3DOM, and a first prototype of the tool for the automatic deployment of 3D content on the web, thus providing basic building blocks for further implementation in the downstream of the V-MUST project. The Virtual Museum testbeds that will be selected might show the urgency to further develop/modify tools, either encompassing single functionalities or services, before the evaluation and in turn will have an influence on the micro projects on which we will decide to focus.

8 Conclusion

This document describes the initial design of the Common Implementation Framework (CIF), based on an analysis of the state of the art tools employed in current Virtual Museum workflows. This analysis has been performed by taking into account the experience of the project partners, their typical workflow and their perceived limitations of current technologies. This document is based also on the output coming from the V-MUST technical questionnaire (presented in the deliverable 4.1) and on several technical meetings about the design of an effective Web-based platform to support the development of the Virtual Museums in the future.

Based on the results of this analysis, we have selected some important tools (mainly visualization tools) and services which are common to the different virtual modeling processes and workflows. We elaborated some first guidelines and ideas for the design and development of the CIF, that shall provide a technological foundation for the V-MUST Platform, together with the V-MUST repository/repositories and the additional services defined in the WP4.

Two state of the art reports, one about user interface design and another one about visualization technologies have been also provided through this document.

References

[ACTIVEX] *Microsoft ActiveX Controls.* http://msdn.microsoft.com

[Bernardini2002] Fausto Bernardini, Holly E. Rushmeier, *"The 3D Model Acquisition Pipeline",* Computer Graphics Forum 21(2): 149-172 (2002)

[Bowman2004] Bowman, D. A., Kruijff, E., LaViola, J. J., & Poupyrev, I. (2004). *3D User Interfaces: Theory and Practice.* Boston: Addison-Wesley.

[Bowman2001] Bowman, D. A., Kruijff, E., LaViola, J. J., & Poupyrev, I. (2001). *An Introduction to 3-D User Interface Design. Presence Teleoperators and Virtual Environments*, 10(1), 96-108. MIT Press. doi:10.1162/105474601750182342

[Bowman1999] Bowman, D. A. , Johnson, D. B. , and Hodges, L. F. (1999). Testbed evaluation of virtual environment interaction techniques, in Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST '99), pp. 26–33.

[Callieri2008] Marco Callieri, Paolo Cignoni, Massimiliano Corsini, Roberto Scopigno, *"Masked photo blending: Mapping dense photographic data set on high-resolution sampled 3D models",* Computers & Graphics 32(3): 464-473 (2008).

[Calori2009] Luigi Calori, Carlo Camporesi, Sofia Pescarin, *"Virtual Rome: a FOSS approach to Web3D"*, Proceedings of the 14th International Conference on 3D Web Technology (Web3D 09), 2009.

[CGAL] *CGAL Project: Cgal, computational geometry algorithms library.* http://www.cgal.org.

[ERGONOMIC] ISO (1998). *ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability*.

[Fanin2011] Bruno Fanin, L. Calori, D. Ferdani, S. Pescarin, *"Interactive 3D Landscapes Online"*, Proceedings of the 3D Virtual Reconstruction and Visualization of Complex Architectures Conference (3D-ARCH 2011), 2-5 March 2011, Trento, Italy.

[Franken2005] Thomas Franken, Matteo Dellepiane, Fabio Ganovelli, Paolo Cignoni, Claudio Montani, Roberto Scopigno: *Minimizing user intervention in registering 2D images to 3D models.* The Visual Computer 21(8-10): 619-628 (2005)

[GLGE] Paul Brunt. *GLGE: WebGL for the lazy.* http://www.glge.org/, 2010.

[GLU] Khronos Group. *GLU OpenGL Utility Library.* http://www.opengl.org/documentation/specs/glu/glu1_3.pdf.

[KHRONOSGROUP] Khronos Group. *Khronos: Open standards for media authoring and acceleration.* http://www.khronos.org, 2009.

[JOGL] *JOGL: Java Binding for the OpenGL API.* http://kenai.com/projects/jogl/pages/Home.

[Nielsen1994] Nielsen, J. (1994). *Heuristic evaluation.* In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods, John Wiley & Sons, New York, NY.

[Norma2010] Norman, D. A. (2010), *Natural User Interfaces are Not Natural*, Interactions, 17(3), 6-10.

[Norma1988] Norman, D. A. (1988). *The Design of Everyday Things. New York:* Doubleday.

[O3D] Google Labs. *O3D.* http://code.google.com/apis/o3d/, 2009.

[OPENMESH] *OpenMesh, visualization and computer graphics library.* http://www.openmesh.org.

[SCENEJS] Lindsay Kay. *SceneJS.* http://www.scenejs.com, 2009.

[Shackel1986] Shackel, B. (1986). *IBM Makes Usability as Important as Functionality.* The Computer Journal, 475-476.

[Shneiderman2010] Shneiderman, B. and Plaisant, C. (2010). *Designing the User Interface: Strategies for Effective Human-Computer Interaction:* Fifth Edition, Addison-Wesley Publ. Co., Reading, MA.

[SPIDERGL] Marco Di Benedetto. *SpiderGL: 3D Graphics for Next-Generation WWW.* http://spidergl.org.

[SPIDERGL] Marco Di Benedetto, PhD Thesis, "Multiresolution Streaming and Visualization of Massive Citiscapes", 2010, University Of Pisa.

[THREEJS] *Three.js A Javascript 3D Engine.* https://github.com/mrdoob/three.js/

[VCGLIB] *VcgLib, Visualization and Computer Graphics library*. http://vcg.sourceforge.net.

[VRML] D. Raggett. *"Extending WWW to support platform independent virtual reality"*. Technical Report, 1995.

[X3D] Leonard Daly, Don Brutzmann. *X3D: Extensible 3D Graphics for Web Authors*. Morgan Kaufmann, 2007.

[WEBGL] Khronos Group. *WebGL - OpenGL ES 2.0 for the Web.* http://www.khronos.org/webgl, 2009.

[WEBGLU] Benjamin DeLillo. *WebGLU: A utility library for working with WebGL .* http://webglu.sourceforge.org/, 2009.

[Wilson2006] Wilson. G, (2006) *Off With Their HUDs!: Rethinking the Heads-Up Display in Console Game Design* [online] Gamasutra.
Available at: http://www.gamasutra.com/view/feature/2538/off_with_their_huds_rethinking_.php [Accessed 10th January 2012]