WebCAME - A 3D Multiresolution Viewer for the Web

Helfried Tschemmernegg
tschemmi@gmx.at

Graz University of Technology
Institute for Computer Graphics and Vision

Abstract

In this document *WebCAME*, a multiresolution viewer for 3D scenes on the internet is described. While being capable of displaying various multiresolution scenes, it has been developed with computer-aided virtual archaeology in mind and will be used primarily to visualize a virtual representation of an excavation site.

Regarding navigation in the scene, a great effort has been made to achieve the best possible usability. By just using the mouse and its buttons, the user can freely move and look around in the scene. This results in a low number of push-buttons being needed, which contributes a lot to simplicity.

Moreover, additional data needs to be displayed sometimes to give further explanations for certain objects. These data are referred to as metadata. To draw the user's attention to them, virtual glass-cases are put over the interesting objects. By clicking one of these glass-cases, the user can change to a special navigation mode that allows to view a single object within a scene by circling around it on an imaginary sphere.

Keywords: 3D, Viewer, Navigation, Metadata, Multiresolution

1 Introduction

This project has been motivated by the *Murale* project that deals with computer-aided virtual archaeology. *Murale* is an abbreviation for *"3D Measurement and Virtual Reconstruction of Ancient Lost Worlds of Europe"*. Led by the Brunel University, London, a consortium of universities and companies engage in the development of reconstruction and visualization tools for use by a team of archaeologists working in Sagalassos, Turkey. The excavation site near Antalya is conducted by a team from the Katholieke Universiteit Leuven which has been excavating the whole area since 1990, led by Prof. Waelkens. Additional Information about *Murale* can be found in [1], [2] and [3]. The homepage (see [4]) may provide some more recent information as well.

As for every virtual reality task, a viewer is needed to visualize the 3D scenes. And to be able to make the scenes accessible for a large group of interested people it is necessary to use the advantages of the internet. Thus, *WebCAME* has been developed as a browser plugin for Mozilla.

There is already a very powerful ISO-standard called VRML (see [5]) that should be perfect for this problem. The following paragraphs explain the reasons, why an extra application has been developed:

Navigation. Many VRML-viewers have been written until the present day. Some of them are available as standalone applications only, others come as plugins for Web browsers. But all of them only have one basic purpose - to display and link together virtual 3D scenes - which they accomplish more or less perfectly. They all have to be designed as multi-purpose viewers, because a wide variety of different scenes may be implemented using VRML. And it is exactly this generality that causes one of the main problems of actual VRML-viewers: It is very easy to get lost in the scene which is mostly caused by somehow unnatural navigation modes. It happens very often that the user finds himself upside-down. This might not be a problem if only a single object is viewed. But it can be very distracting, if that happens in the 3D model of a real scene where the observing visitor is supposed to stand on the ground and watch buildings and other objects. In VRML, specifying the so called WALK-mode is one possible solution. In this mode the user is always in an upright position. Furthermore, gravity ensures that the user always stands on the ground. But if an unexperienced user activates the FLY-mode, which is also a standard navigation mode of VRML, it is again very easy to lose orientation. Although it is no problem to restrict the selectable navigation modes, this is not done in most of the VRML files available on the internet, since the FLY-mode may of course be useful in many cases. Another solution is to provide the user with viewing points. In the case of

getting lost he can choose from these points and will find himself in a stable, upright position again. But this solution fights the symptoms instead of making it impossible to get lost. So a better solution had to be found and it has been decided to give *WebCAME* a navigation that comes as close as possible to that in real life. There is no way for the user to get into any upside-down position.

Metadata. Another important requirement was the visualization of metadata. While this is extremely important for the intended application area of virtual archaeology, it can also be of great use for visualizing cities and their sights or any other 3D scene that contains interesting objects.

All objects and artifacts that are found on the Sagalassos excavation site are being reconstructed digitally as textured 3D models by the team of the *Murale* project. But the archaeologists working at the excavation site and in the laboratories do not only attempt to reconstruct and preserve the artifacts. They also discover a huge quantity of non-visible facts about every single object that is found. These facts are then stored in the central *Murale* database where they can be accessed for further investigation. If the 3D models of those objects and artifacts are displayed in a virtual representation of an excavation site, it is important to show the user, that further information exists for certain objects. If he is interested in such information, it must be easy to obtain, for example by clicking on the desired object. The task of drawing the user's attention to an interesting object has been accomplished by wrapping it with its own bounding box. To provide the object's visibility within those boxes anyway, they are overlaid using alpha-blending. This technique has been implemented in its own so-called *HotSpot*-class to allow portability to other applications.

Multiresolution. Finally, when displaying the highly detailed scenes on normal PCs, even accelerated graphic cards do not have the necessary capacity to display smooth animations (for an example, see [6]). This problem can be solved by simplifying the scene with efficient multiresolution algorithms. But an efficient implementation of those algorithms is nearly impossible with VRML. In fact, this is the main reason that prevents VRML from being used for the *Murale* project. CAME (described in [7]) is a multiresolution system that supports progressive transmission of highly detailed polygon meshes. This is very useful, especially if the 3D data has to be transmitted over the internet. It allows to first display a rough model of the scene with virtually no delay for the user. This rough model can then be refined depending on the available bandwidth and the actual viewing direction.

1.1 Our Approach

There are some viewers supporting multiresolution already. One of them is MetaStream^{TM} by Viewpoint Company, which is described in [8]. It is capable of streaming multiresolution meshes and progressive transmission, which is perfect for transmitting huge polygon meshes. So, it would fully satisfy the requirement for multiresolution capability. But for the intended prime application field of *WebCAME*, which lies in virtual archaeology, the visualization of metadata is another very important feature. A lot of non visible additional data are collected in a huge database. And there must be a fully integrated capability to access this database from the 3D scene viewer. Thus, object identifiers are added to the multiresolution data as described in [9]. The object identifiers can be used for another important feature: WebCAME will be able to unhinge single objects from the whole multiresolution hierarchy. Thus, the user will be able to manipulate single objects that are actually part of the multiresolution mesh. For these reasons *WebCAME*, a viewer covering all those desired abilities has been developed in this project.

1.2 Overview

A detailed description of the implemented components will be given in the following sections. In section 3, first the specifications of *WebCAME* are presented. The following sections describe the methods used for navigation, visualization of metadata and multiresolution. Section 4 explains the implementation-details of the methods described in section 3. Section 5 covers the test data that were used while developing the viewer. The results are presented in section 6 with some screenshots. Finally, section 7 gives a future outlook to features that are planned to be implemented in the near future.

2 Related Work

2.1 User Interface

The design of user-interfaces is a very common task in programming. Various publications about optimizing the usability of programs can be found. With the increasing performance of accelerated 3D graphics hardware and the resulting high number of viewers for virtual 3D environments, the question on the usability of those viewers emerges. Some very interesting surveys and ideas about this topic can be found in [10] and [11].

2.2 Spatial Navigation

2.2.1 Moving Freely

The task of moving within a virtual scene has already been treated in many publications. An interesting survey about different ways of navigation was conducted in [12]. In [13] a toolset for navigation in virtual environments is described. It has been developed with focusing on principles of navigation in the real world.

2.2.2 Object-Centered Navigation

Object-Centered Navigation is a bit more specialized than moving around freely. Some center must be chosen, around which the user can move in this navigation manner. Although it may not be a very common mode of navigation, there are several publications available dealing with this topic. One proposal is presented in [14]. The method of raycasting to select objects in a scene is introduced in [12]. A similar method has been used to check whether the user has hit or failed one of the *HotSpots* with a mouse click.

2.3 Multiresolution

To enable *WebCAME* of being used on the internet, the bandwidth required to transmit a scene's 3D data must be kept in mind. Multiresolution techniques are a very good way to allow both, reasonable scene-loading times and the ability to refine the scene successively according to the available bandwidth. Thus, *CAME* is used for this purpose. A detailed description of *CAME* can be found in [7]. An application of *CAME* technology in web context is described in [15].

3 Specifications

As stated in the introduction, *WebCAME* has three main advantages over conventional 3D-scene viewers. First, the navigation is less general which results in a more natural handling, bearing surface based exploration in mind. Second, it is equipped with a good possibility to direct the user's attention to objects with further background information in the form of *HotSpots*. And third, the user can start exploring a scene immediately due to *CAME* that provides support for progressive multiresolution meshes.

3.1 Navigation

Navigating in virtual environments is not always easy for the user. The main problem is, that usually there is only a 2D-pointing device (mouse, touchpad, trackball, etc.) to navigate in a 3D scene that is displayed with 2D display technology. That can be quite distracting for human beings who are used to live in a fully-fledged 3D environment. Although the mainly focused user group for the *WebCAME* viewer are archaeologists, it might also be useful for museums to present the excavation site to the visitors or even for a presentation on the internet. That was the reason why an easily learnable navigation had to be invented. The first versions of *WebCAME* had many pushbuttons for choosing the actual navigation modes. The current version comes with only two pushbuttons. This is possible because all available mouse buttons (usually two or three) are used for navigating and moving within the scene. This approach has emerged as the fastest and easiest way of navigation.

As with every program, the user must learn how to operate the *WebCAME* viewer, to get the desired results. To keep this learning process as easy as possible, existing knowledge of how to use a computer mouse is used. Every user knows that a mouse click is mostly followed by some action happening on the display. And the commonly used drag-and-drop mechanism is a perfect analogue to real life where you can grab an object with the hand and put it somewhere or turn it around while viewing it. Furthermore different mouse buttons usually have different meanings. The left button is the one that is used for the main actions while the right button very often leads to optional functions. Most of the computer mice also provide a third button. This button is also used for navigation. But to enable users with 2-button mice to use all functions as well, pressing the middle button is equivalent to pressing both, the right and the left button, at the same time.

According to these facts, a well-elaborated navigation mode has been developed. There are two different camera-modes, the *Free-Camera-Mode* and the *Object-Centered-Mode*. In both modes, there is a direct interaction of mouse-movement and in-scene-movement. That means that the virtual camera does will move if the user does not move the mouse. This is another difference to common viewers, that mostly continue to move the virtual camera in a speed according to the distance between the clicking point and the actual position of the mouse-pointer. This mode of navigation increases the risk of getting lost in the scene and is therefore not used for *WebCAME*.

3.1.1 Free-Camera-Mode

In *Free-Camera-Mode* the left mouse button is assigned the most important ability of looking around in the scene. While pressing the left mouse button, the mouse movement is captured to turn the camera. Any movement in x-direction changes the camera's panning-angle, which corresponds to looking left and right in reality. Movements in y-direction lead to a change of the tilt-angle, which would mean to look up and down in real life.

Pressing the right mouse-button provides another analog that is similar to walking forward, backward and sidewards in reality. Mouse-movement in x-direction moves the camera sidewards while moving the mouse in y-direction moves the camera forward and backward now.

The third possibility for camera movement is chosen by pressing the middle mouse button or alternatively both, the right and the left button simultaneously. This mode allows the user to change the altitude with movements in y-direction. Movements in x-direction again lead to panning the camera left and right.

Table 1 gives an overview of the possible camera movements in *Free-Camera-Mode*.

Mouse-Button	*Action*
left	look around x: look left and right y: look up and down
right	move & look x: move left and right y: move forward and backward
left & right (middle)	change altitude, move sidewards x: look left and right y: move up and down

Table 1: Behaviour in *Free-Camera-Mode*

3.1.2 Object-Centered-Mode

Viewing objects in a scene can be annoying without a proper camera mode that supports circling around the object with the camera always pointing to that object. Thus, a special camera mode, which accomplishes this task has been developed for *WebCAME*.

The *Object-Centered-Mode* of *WebCAME* is activated automatically by choosing any *HotSpot* with a mouse-click on it. Thus, the *HotSpots* have to be displayed first. To do this, the user must click on the *Choose Object* Button, which displays all *HotSpots* within the current scene. If the user then chooses any object, the camera will automatically move close to that object with its viewing direction pointing to the object's center. Now the user can move on an imaginary sphere surrounding the object. The angles of the camera are set automatically to point its aiming always to the center of the object, while *Object-Centered-Mode* is being used.

By pressing the left mouse-button and moving the mouse, the user can move around freely on the previously described imaginary sphere. Movement in x-direction means walking around the object on a circle lying parallel to the ground-plane while moving the mouse in y-direction moves the camera up or down on the imaginary sphere.

With the right mouse-button, the distance to the object can be changed according to mouse movement in y-direction. Moving in x-direction again allows circling around the object.

Table 2 gives an overview of the possible camera movements in *Object-Centered-Mode*.

Mouse-Button	*Action*
left	x: circle around object y: move up or down on the virtual sphere
right	x: circle around object y: decrease or increase the distance to the object

Table 2: Behaviour in *Object-Centered-Mode*

3.2 Visualization of Metadata

Every artifact that is found on an excavation site provides many pieces of information in addition to its bare visual appearance. Archaeologists usually discover a huge number of facts that are not visible like age, material, possible purpose, level of confidence etc. All those facts are usually collected in the central database of the *Murale* project. If a virtual 3D scene is constructed from this database, the user can walk around and explore the scene. But how does he know that there is further interesting information available for certain objects?

Thus, a way must be found how interesting objects can attract the user's attention. For *WebCAME*, this is done by displaying virtual glass cases enclosing the interesting objects. These glass cases are being displayed by the *HotSpots*-class. By clicking the *Choose Object* button, the user can activate the display of those *HotSpots*. A second click on the same button will switch back to normal scene-observation mode without *HotSpots* being displayed. If the user then decides to have a closer look on any object that is marked by a virtual glass case, he can select it by a single mouse-click. The *WebCAME* viewer switches to *Object-Centered-Mode* and moves the camera close to the object. Then, the user can explore the object. To leave the *Object Mode*, the user must only click the *Free Camera*-button and the camera may be moved freely again.

Furthermore, additional information can be displayed in another frame of the webpage using HTML containing text, pictures or even videos, for example. This can easily be done because *WebCAME* supports calling *JavaScript* functions and means a great deal of freedom in the design of multimedia web presentations about archaeological excavation sites or any other online resource using *WebCAME*.

3.3 Other Features

Besides the previously described camera modes and visualization functions, three further features have been implemented in *WebCAME*:

- a wireframe mode, which can be useful if textures are applied to multi-resolution meshes. Because of the textures it can be impossible to recognize the exact edges of the faces in the scene. But for developing multiresolution environments it is quite important to know where the edges are. Furthermore it has been a useful feature for visualizing the structure of the ground surface while using the simple test-dataset described below.

- a possibility to switch textures on or off. During the development phase it can be very informative to see the untextured version that reveals the construction details of every object. But for a realistic appearance of scenes, textures are a must. So, the use of textures can be switched on or off easily to satisfy both requirements. Those of developers who want take a look at the 'backstage-area' of a scene and those of viewers who want to view realistic scenes.

- a slider to choose the desired level of detail. It would not make any sense to transmit more data than the user wants to see. Thus there is the possibility to first view a low-detailed version of a scene (slider on the left side). If further detail is desired, the level of detail can be raised by moving the slider to the right.

These features can be controlled by three additional widgets: two checkboxes and one slider. The first checkbox is used to enable or disable the wireframe-mode. The second checkbox triggers the application of textures in the scene. And the third control widget, the slider, is there for choosing the desired level of detail.

4 Implementation

4.1 Environment

WebCAME has been developed and implemented using GNU Linux, the Qt library by Trolltech [16] and the Plugin-SDK by Netscape [17]. For further information about Qt's support for Netscape/Mozilla plugins refer to the *LiveConnect Plugin* site [18] by Trolltech.

4.2 Framework

A basic code-framework for Netscape browser-plugins, which has been programmed by Markus Grabner, was the starting point for my work. This framework provided the possibility to compile the whole project both as a standalone application and a browser-plugin, which is made possible by the modular concept of Qt's QWidget class. Every QWidget can contain other QWidgets. So there are two containers: one for the standalone application and the other one for the plugin. Both of them contain the same viewer widget. The viewer plugin itself was based upon the QGLWidget class, which provides *OpenGL* support for Qt widgets. Of course this is essential for a 3D viewer. For the development and debugging process, the standalone application was used most of the time, because it is more convenient to call a single application than always having to use a browser to examine the results of a build.

4.3 Scene Visualization

To render the virtual scene on screen, *OpenGL* is used. The reason for this choice is simple: *OpenGL* is the best solution for writing portable software that uses 3D acceleration. The *OpenGL Programming Guide* [19] has been a great help for that task. *WebCAME* has been developed under Linux, but there are plans to port the viewer to Microsoft Windows platforms to be used as ActiveX plugin for Microsoft Internet Explorer. Although Microsoft has developed their own standard named Direct3D the Windows-Version will also use *OpenGL*.

4.4 Navigation

All navigation purposes can be accomplished with very basic mathematical knowledge about vectors. The first step is to construct a virtual camera, whose position and aiming is known to all components involved in navigation. For this purpose, a vector with three components is used to store the current position of the camera in the scene. Then, there are two float-variables that store two angles defining the camera's aiming. One of them, the panning angle, holds values between 0 and 2π, indicating to which direction parallel to the ground surface the camera is pointing. The other one is the pitch-angle and holds values between $-\pi/2$ and $+\pi/2$ to indicate if the camera is looking up or down. Before we can start to render anything, the transformation matrices must be set according to the position (translation) and aiming (rotation). of the virtual camera. The required values for these operations can be easily obtained from the data known about the camera.

4.4.1 Free-Camera-Mode

The translation operations needed for moving the camera are very easy to understand. For moving for-

ward, a vector has to be added to the camera's actual position. This vector is calculated from the panning-angle only. For moving sidewards, another vector is needed. It can be obtained simply by adding $\pi/2$ to the panning angle. Moving up and down means increasing or decreasing the y-value of the camera's position. The y-values are changeable freely, because there is no terrain tracking with collision detection yet.

For looking around, the two angles indicating the camera's aiming are responsible. Looking left and right is done by increasing or decreasing the panning-angle. The pitch-angle must be increased or decreased in order to look up and down.

4.4.2 Object-Centered-Mode

Camera movement in *Object-Centered-Mode* first seems to be a little more tricky. But after having a closer look it is quite easy as well.

Circling Around The Object. To move the camera around the object on an imaginary circle, first, the panning angle is increased or decreased. Then a vector is calculated that points to the inverse direction of the camera. This inverse direction can be easily obtained by adding π to the panning-angle. To update the camera's position, the vector that has just been calculated must be added to the center of the virtual circle.

Moving Up And Down. Moving up and down is very similar to circling around the object. Here, the pitch-angle is used instead of the panning angle. And there is the additional requirement that the angle must be between $-\pi/2$ and $+\pi/2$ to prevent any upside-down-situations. In fact, the user is detained from going down the other side of the imaginary sphere, once he has reached the top. The same principle applies for the bottom.

4.5 Visualization of Metadata

As described in section 3, the visualization of metadata is done by bounding boxes surrounding the objects like glass cases. These glass cases are displayed using alpha-blending which is supported by *OpenGL*. There is one aspect that must be respected. In the *OpenGL* rendering-pipeline, alpha blending is done as part of the per-fragment operations. These operations are the last ones being executed before the pixels are written to the appropriate buffer. When painting the alpha-blended surfaces, *OpenGL* does not take care of their depth values. If they are inserted in the rendering pipeline in the wrong order, this will lead to an unrealistic output because an alpha-blended surface that is rendered after another one with a smaller depth will not be transparent. However it is easy to avoid this problem if the programmer takes care of the correct rendering order by himself. Thus, the blended surfaces of the bounding boxes must be sorted, according to their depth values before inserting them into the rendering pipeline.

The *HotSpots*-class is responsible for the management and visualization of the glass cases. It holds some variables that are important to uniquely define the position and the size of the bounding boxes. To be able to obtain any additional information stored about a *HotSpot* in a scene, a unique object-identifier is needed as well. Additional information about how to provide such object-identifiers in multiresolution meshes can be found in [9].

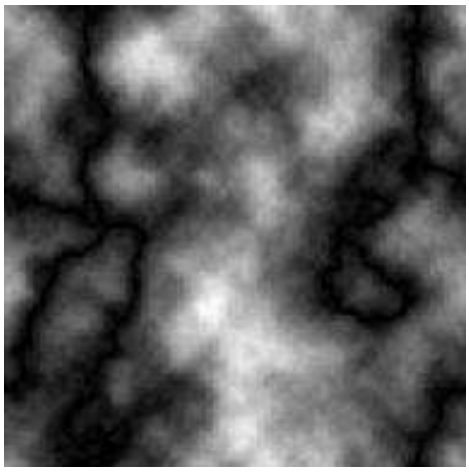
With the knowledge of these data it is possible to build a bounding box very easily, although the use of alpha-blending makes this task a little difficult. To prevent problems with the order of rendering, an additional mechanism is needed, as it has been described before. Moreover an interesting discovery can be made. By looking at a box it is obvious that a maximum of 3 faces can be seen at once. In other cases 2 faces or even only 1 face can be seen. And it is sufficient to render only the visible faces. But to determine which faces are visible by the camera, the camera-position is needed as well. So, the position of the camera must be handed over when issuing the render function. Another reason for doing this is the fact, that the blended surfaces must be rendered in the correct order, which can be achieved by sorting them according to their distance to the camera. With this additional knowledge, it is no problem to draw the blended faces.

5 Test Data

The development of the navigation and metadata capability of the *WebCAME*-viewer happened independently from the development of *CAME*. Therefore a very simple and small dataset has been used during the development phase. So, the functionality of *WebCAME* could be tested independently from the multiresolution database.

A textured triangle mesh representing a landscape was created. The elevation data was obtained from a heightmap grayscale picture with the size of 192×192 pixels (see in Figure 1(a)). Every pixel represents a certain height according to its gray-value. The brighter a pixel, the larger the vertex' height value must be. The triangle mesh is then overlaid with the appropriate texture which has the same size of 192×192 pixels (see Figure 1(b)).

To test the *HotSpots*-Class, two simple objects have been designed and stored as .ply-files. This file format has the advantage of being easily changeable with



(a) Heightmap used to create the ground surface mesh

(b) Texture used for the ground surface

Figure 1: Heightmap and Texture used for the Terrain

(a) Cross on a summit

(b) Uhrturm on Schlossberg in Graz, Austria

Figure 2: The two objects used for testing

a normal editor without having to use a special program. It can be used to store indexed face sets that may also contain texture information.

Figure 2(a) shows a cross as it is often seen on summits and Figure 2(b) shows a model of the Uhrturm (a landmark on Schlossberg in Graz, Austria).

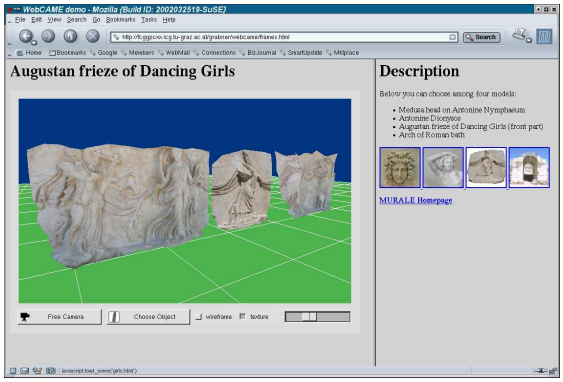
6 Results

In this section, some screenshots will be presented. Figures 3 and 4 show *WebCAME* as web browser plugin. Figures 5 and 6 present a look at the standalone version which was used in the development phase.

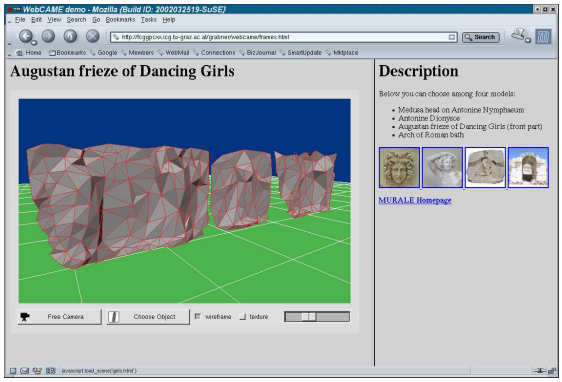
6.1 WebCAME as Browser Plugin

One of the primary goals in the development of *WebCAME* was to implement it as browser plugin. This has the advantage of being able to distribute a lot of multimedia information quite easily by using existing internet technology. In the following screenshots, the rendering window shows original figures from the *Murale* database. The surrounding HTML-pages demonstrate how the plugin can be embedded into an existing information system.

Figure 3(a) shows a basic web-page displaying a 3D view of some artifacts in the left frame and further explanations in the right frame. The thumbnail images on the right side can be used to choose the object that is shown in the left frame. Figure 3(b) demonstrates the use of the wireframe mode combined with the ability to switch off textures. The difference between the



(a) Possible design of a webpage displaying HTML information on the right side and the *WebCAME* plugin to the left side.



(b) The same screenshot as Figure 3(a) but with textures turned off and wireframe mode turned on.

Figure 3: Two screenshots of *WebCAME* embedded into a webpage as browser plugin

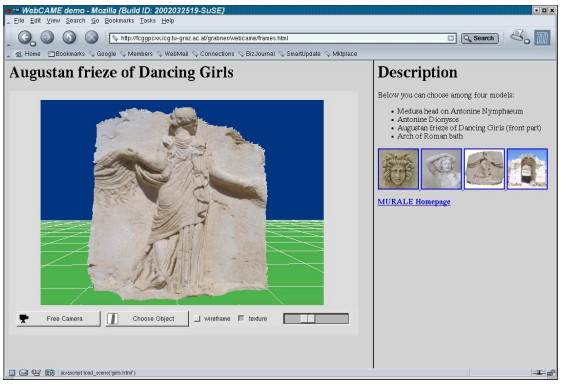


Figure 4: A detailed view of an artifact.

previous two examples shows the immense amount of reality that is added to a scene by applying textures to a relatively simple triangle mesh. In Figure 4, a more detailed view of another artifact is presented.

6.2 WebCAME as Standalone Application

During the development phase it was important to be able to test the viewer independently from the multiresolution database. Thus, as mentioned before, a set of test data has been created to easily demonstrate the viewer's abilities. Figure 5(a) presents a first look to the scene with both test-objects visible. In Figure 5(b), the Button "Choose Object" has been pressed to show the *HotSpots* in the Scene. Thus, the previously described virtual glass-cases are put over the two test-objects. By clicking on the box around the Uhrturm, which is located next to the user's position, the *Object-Centered-Mode* is enabled. Figure 6(a) shows a view from this mode, where the user can move around the object on the surface of the imaginary sphere that has been described previously. Finally, Figure 6(b) shows an overview of the scene with activated wireframe mode and without textures.

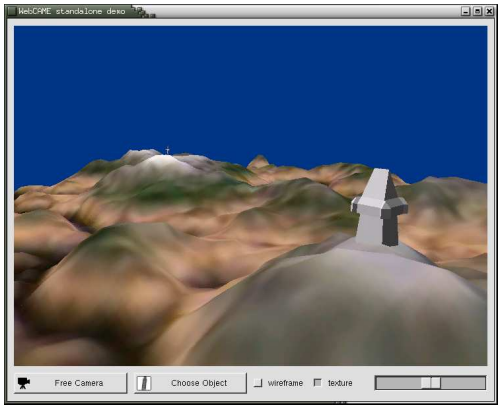
7 Outlook

A status window will be integrated into the rendering window. It can be used to assist the user by displaying additional information. For example, hints about how to navigate can be given. The information will be overlaid using alpha-blending and will look like the Head Up Displays (HUDs) known from many games. Another interesting possibility is to assist the user with orienting by displaying a small map similar to the one presented in [20] and [21].

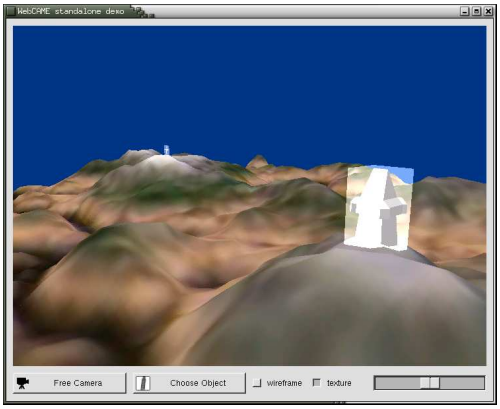
Until now, there is not yet any collision detection or terrain tracking. These features will be implemented in one of the future versions. An algorithm will have to be designed to take care of updated geometry information, because the geometry is changed continuously while viewing the scene. Object identifiers as described in [9] will be used to differentiate between terrain information and objects.

Another feature will be an intelligent automatic speed control. The navigation speed thereby will be set to appropriate values, depending on the surrounding geometry. This approach has been chosen to prevent the user from getting confused by too many degrees of freedom.

It will be possible to grab an object for example a vase or a piece of rock to examine it more closely. The user may turn it around as if he would hold it in his hands. This will be realized by an additional "navigation mode". It will be implemented using the Arcball technology developed by Ken Shoemake (see [22]). This technology has proven to be the most intuitive one, concerning object orientation, in several usability studies, for example in [23] or in [24].

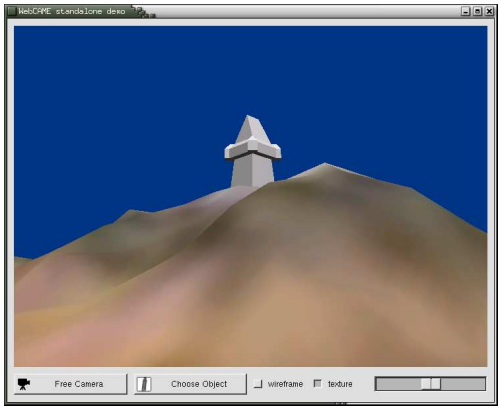


(a) A screenshot of the standalone application, which was used during the development phase, showing the two test-objects.

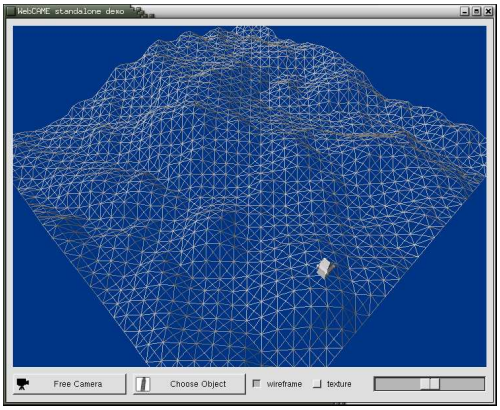


(b) The same view as in Figure 5(a) but with *HotSpots* turned on.

Figure 5: Two screenshots showing how interesting objects are indicated



(a) A screenshot that was made while in *Object-Centered-Mode*.



(b) An overview of the scene in wireframe-mode which shows the triangle mesh of the ground-surface.

Figure 6: Another two screenshots of the standalone application

Finally, it is planned to port the viewer to ActiveX technology to support Microsoft Internet Explorer. This way of action has been chosen because IE is the most commonly used web-browser, according to statistics.

References

[1] J. Cosmas, T. Itegaki, D. Green, E. Grabczewski, F. Weimer, L. Van Gool, A. Zalesny, F. L. Desi Vanrintel, M. Grabner, K. Schindler, K. Karner, M. Gervautz, S. Hynst, M. Waelkens, M. Pollefeys, R. DeGeest, R. Sablatnig, and M. Kampel, "3D MURALE: A Multimedia System for Archaeology," in *Proceedings ACM Virtual Reality, Archaeology and Cultural Heritage (VAST 2001)*, November 2001.

[2] R. Sablatnig and M. Kampel, "The virtual reconstruction of an archaeological site - an overview of the MURALE project," in *Proceedings of the 6th Computer Vision Winter Workshop 2001* (B. Likar, ed.), (Bled, Slovenia), pp. 60–70, Slovenian Pattern Recognition Society, February 2001.

[3] J. Cosmas, T. Itegaki, D. Green, E. Grabczewski, F. Weimer, L. Van Gool, A. Zalesny, F. L. Desi Vanrintel, M. Grabner, K. Schindler, K. Karner, M. Gervautz, S. Hynst, M. Waelkens, M. Pollefeys, R. DeGeest, R. Sablatnig, and M. Kampel, "A Novel Multimedia System for Archaeology," in *Proceedings of Forum International de Musees*, (Museum National d'Histoire Naturelle, Paris, France), April 2002.

[4] Brunel University, London, "Homepage of the 3D Murale Project." (http://www.brunel.ac.uk/project/murale/).

[5] The Web 3D Consortium, "The Virtual Reality Modeling Language: International Standard ISO/IEC 14772-1:1997," 1997. (http://www.web3d.org/).

[6] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk, "The digital michelangelo project: 3D scanning of large statues," in *Siggraph 2000, Computer Graphics Proceedings* (K. Akeley, ed.), pp. 131–144, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[7] M. Grabner, "Compressed adaptive multiresolution encoding," *Journal of WSCG*, vol. 10, pp. 195–202, Feb. 2002. ISSN 1213-6972.

[8] V. Abadjev, M. del Rosario, A. Lebedev, A. Migdal, and V. Paskhaver, "Metastream," in *Proceedings of the fourth symposium on Virtual reality modeling language*, (Paderborn, Germany), pp. 53–62, ACM Press, 1999. ISBN 1-58113-079-1.

[9] M. Grabner and H. Tschemmernegg, "Object identification in compressed view-dependent multiresolution meshes," *submitted for publication*, 2003.

[10] D. S. Tan, G. G. Robertson, and M. Czerwinski, "Exploring 3d navigation: combining speed-coupled flying with orbiting," in *CHI*, pp. 418–425, 2001.

[11] F. Pittarello and A. Celentano, "A Multimodal Approach for Orientation and Navigation in 3D Scenes," tech. rep., Universita Ca Foscari, Dipartimento di Informatica, 1999.

[12] K. Hinckley, R. Pausch, J. C. Goble, and N. F. Kassell, "A survey of design issues in spatial input," in *Proceedings of the 7th annual ACM symposium on User interface software and technology*, (Marina del Rey, California, United States), pp. 213–222, ACM Press, 1994. ISBN 0-89791-657-3.

[13] R. P. Darken and J. L. Silbert, "A Toolset for Navigation in Virtual Environments," in *Proceedings of ACM User Interface Software & Technology*, 1993.

[14] C. Elcacho, T. Dingel, and R. Klein, "Object-centered navigation in virtual construction applications," in *Proceedings of European Association for Computer Graphics WSCG* (V. Skala, ed.), pp. 33–40, 2001.

[15] M. Grabner, "WebCAME: A Light-weight Modular Client/Server Multiresolution Rendering System," in *Web3D 2003 Symposium Proceedings* (S. E. Spencer, ed.), (Saint Malo, France), pp. 63–72, Mar. 2003. ISBN 1-58113-644-7.

[16] Trolltech, "Qt Reference Documentation." (http://doc.trolltech.com).

[17] Netscape, "Netscape Gecko Plug-in API." (http://devedge.netscape.com/library/manuals/2002/plugin/1.0/).

[18] Trolltech, "Qt-based LiveConnect Plugins." (http://doc.trolltech.com/3.1/nsplugin.html).

[19] M. Woo, J. Neider, T. Davis, and D. Shreiner, *OpenGL Programming Guide*. Addison-Wesley, 3rd ed., 1999. ISBN 0-201-60458-2.

[20] Y. Kitamura, S. Fukatsu, T. Masaki, and F. Kishino, "Intuitive Navigation in an Enormous Virtual Environment," in *Proceedings of the International Conference on Artificial Reality and Tele-Experience (ICAT)*, (Tokyo, Japan), pp. 163–169, December 1998.

[21] S. Fukatsu, Y. Kitamura, T. Masaki, and F. Kishino, "Intuitive control of "bird's eye" overview images for navigation in an enormous virtual environment," in *Proceedings of the ACM symposium on Virtual reality software and technology 1998*, (Taipei, Taiwan), pp. 67–76, ACM Press, 1998. ISBN 1-58113-019-8.

[22] K. Shoemake, "ARCBALL: A user interface for specifying three-dimensional orientation using a mouse," in *Proceedings of the Graphics Interface '92*, (Vancouver, Canada), pp. 151–156, 1992.

[23] K. Hinckley, J. Tullio, R. Pausch, D. Proffitt, and N. Kassell, "Usability analysis of 3d rotation techniques," in *Proceedings of the 10th annual ACM symposium on User interface software and technology*, (Banff, Alberta, Canada), pp. 1–10, ACM Press, 1997. ISBN 0-89791-881-9.

[24] F. Ritter, "Interaktives Zusammensetzen von 3D-Modellen zur Unterstuetzung des raeumlichen Verstaendnisses," Master's thesis, Otto-von-Guericke Universitaet Magdeburg, 1999.