VIRTUAL ART GALLERY TOOL

Pedro Miguel Semião and Maria Beatriz Carmo
Faculdade de Ciências da Universidade de Lisboa, Campo Grande, 1749-016 Lisboa, Portugal
psemiao@gmail.com, bc@di.fc.ul.pt

Keywords: Virtual Gallery, e-Exhibitions, X3D, Xj3D.

Abstract: This paper describes a set of two applications comprising the Virtual Art Gallery tool: the Space Picker application and the Virtual Exhibition Builder application. The purpose of this tool is to allow users to interactively create a virtual exhibition of artworks in a pre-built virtual model. We use X3D for the models and Java with Xj3D for the display and handling. The works of art are handled through a MySQL database. The solution is entirely based in free software.

1 INTRODUCTION

There are a large number of museums and art galleries willing to share their exhibitions online, since they realize it is not possible for all interested people to visit them. These exhibitions usually display scans of the works of art in an HTML page. Some go a bit further, like the Louvre Museum (httpLouvre, 2007), using QuickTime VR (QTVR) movies (httpQTVR, 2007) that allow a panoramic view of a room from a single spot. This is done in an attempt to bring the virtual visiting experience closer to the real one. QTVR is very limited, not being based on models but on panoramic pictures. In contrast, VRML and X3D (httpWeb3d, 2007) provide the description of 3D models allowing for a rich medium in which one could base exhibitions that are closer to reality. The task of creating the model of a building can be achieved, in a matter of weeks by a trained professional, with CAD tools and a blueprint. As for the exhibition of art works in the created model, it would be necessary to change the model each time one wishes to have new content. As is mentioned in (Wojciechowski, 2004), "The work on setting up an exhibition should be performed by museum staff (e.g., curators), which cannot be expected to be IT experts". Our tool will allow for people with no specific training to create a new art exhibition in a previously made X3D model.

In the remainder of this paper we will present work related to virtual exhibitions in Section 2 and a brief view of the tool in Section 3. In Section 4 we present one of the applications that comprise the tool, the Space Picker, and in Section 5 we present the main application, the Virtual Exhibition Builder. In Section 6 we present ongoing work and some prospects for future work and in Section 7 we draw some conclusions from our work.

2 RELATED WORK

Despite the almost ubiquitous presence of the Internet and the growing popularity of virtual visits, there is still not much work being done in the area of bringing these technologies to a level of usability and affordability, in terms of organizing a virtual exhibition.

In Virtual Art Gallery (Hrk, 2001), the user can create a simple VRML 3D model and attach paintings to the walls. Since the modeling part of the application is very limited, it is not possible to produce quality models.

Taking a different approach, The Virtual Showcase (Bimber, 2001) aims to merge the display of real and virtual artifacts in the real location, using 3D virtual augmentation.

ARCO (httpArco, 2007) is a platform with a very broad scope, ranging from tools for 3D information acquisition to the organization and display of collections. The visualization of the virtual representation of museums is achieved through the use of X-VRML (Walczak, 2003), a high-level XML-based procedural language that adds dynamic modeling capabilities to virtual scene description standards. Mixed reality presentations are also pro-

vided (Walczak, 2005). These presentations can be created through a user-friendly content management application that uses templates with encoded visualization and interaction rules.

3 VIRTUAL ART GALLERY TOOL

The Virtual Art Gallery tool was designed having some particular goals in mind: it should only use free software and the user is not required to have any knowledge of X3D to use the software, allowing for a broader user base. The tool is comprised of two applications. The first, Space Picker, allows the user to choose the surfaces that will be available for art display in the X3D model. The application also makes some changes in the X3D file, making it compatible with the second application. The second and main application, Virtual Exhibition Builder, permits the user to add paintings to the available surfaces, chosen in the first application. The applications were developed in Java. Xj3D's libraries were used for the display and interaction with the X3D model. The task of painting selection is handled with a MySQL database support.

4 SPACE PICKER APPLICATION

The Space Picker application readies the X3D file so that it can be used with the Virtual Exhibition Builder application. It also gives the user the possibility to choose several surfaces on the model. These surfaces will be available in the Virtual Exhibition Builder application as possible targets for artwork display. To be able to interact with any surface in X3D there must be a sensor placed on the surface. The application counters this by creating a temporary X3D scene with sensors on every surface. This temporary scene is then loaded through an Xj3D browser, where the user will select the surfaces where expositions will be allowed. When the user is satisfied with the results he can save a new X3D file.

This file will have sensors on the chosen surfaces and will be compatible with the Virtual Exhibition Builder application.

A short description of the dataflow, as seen in Fig. 1 and the general workflow of the application follow.

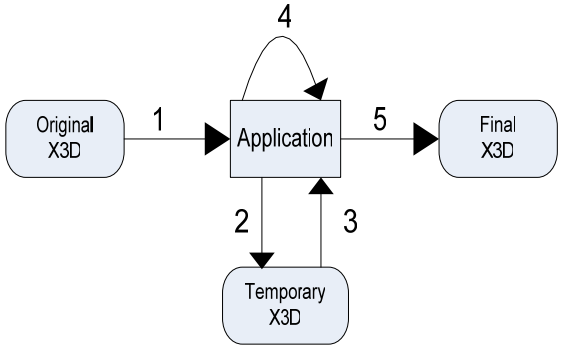

Figure 1: Space Picker Data Flow.

When the Original X3D is loaded (1), the code is stored in two StringBuffers, named Pre and Pos, and one Vector of StringBuffers, named Shapes:

- Pre: contains all the code until the first shape creation.
- Shapes: contains one shape or transformation *per* StringBuffer.
- Pos: contains all the X3D code after the last shape definition.

This will provide individual shape code access to and control of the application. The Temporary X3D file is created (2) from these three elements, making the necessary adjustments, as in the following sequence:

1. Pre is sent to file unaltered.
2. A new material definition to identify the selected surfaces is sent to file.
3. Each shape is sent to file inside a new sensor. The sensors are named SensorN, where N is a unique sensor number. The appearance tag is also named as SensorAN, with the same N, to allow access to the appearance's properties by the application.
4. Pos is sent to file unaltered.

The newly created Temporary X3D file is then loaded into an Xj3D browser (3). The application keeps spatial information about all the sensors as well as their selection status. This information is stored in a vector of instances of the Sensor class, having a direct correspondence with the Shapes Vector.

Surface selection is achieved through Xj3D's SAI (Scene Access Interface). User selected surfaces will be presented in red, indicating their selection status, as seen in Fig. 2. This is achieved by a change in the surfaces material definition. If an already selected surface is activated it will revert to its former material definition, which was stored

when the first change was made, clearing the selected status.

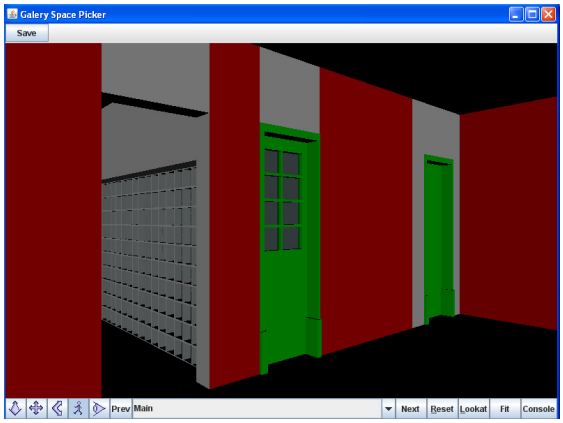

Figure 2: Space Picker Screenshot.

When the user chooses to save his work the Final X3D file will be created, in the following manner:

1. Pre is sent to file unaltered.
2. For each shape in Shapes it is verified if the corresponding sensor is selected.
   a. If it is not, the shape is written to file as is.
   b. If it is, the shape is inserted into a group with a sensor, named SensorN, N being a unique number starting with one. The Coordinate and IndexedFaceSet tags are also named in a similar way, to allow access from the Virtual Exhibition Builder application.
3. Pos is sent to file unaltered.

The file is then ready for use with the Virtual Exhibition Builder application.

5 VIRTUAL EXHIBITION BUILDER APPLICATION

The Virtual Exhibition Builder application builds the virtual exhibition, using the X3D file that was modified by the Space Picker application and the paintings in the database, as seen in the application layout in Fig. 3.

In section 5.1 we present the application's simplified class diagram along with a brief explanation of each class' responsibilities. In section 5.2 we present our method of handling the X3D code and in section 5.3 the display and manipulation of the X3D scene. In section 5.4 we present the application from the user's perspective.

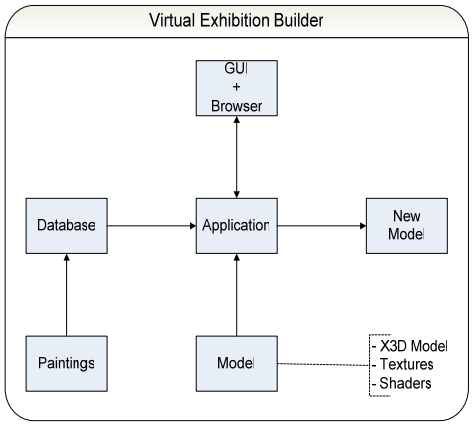

Figure 3: Virtual Exhibition Builder Application Layout.

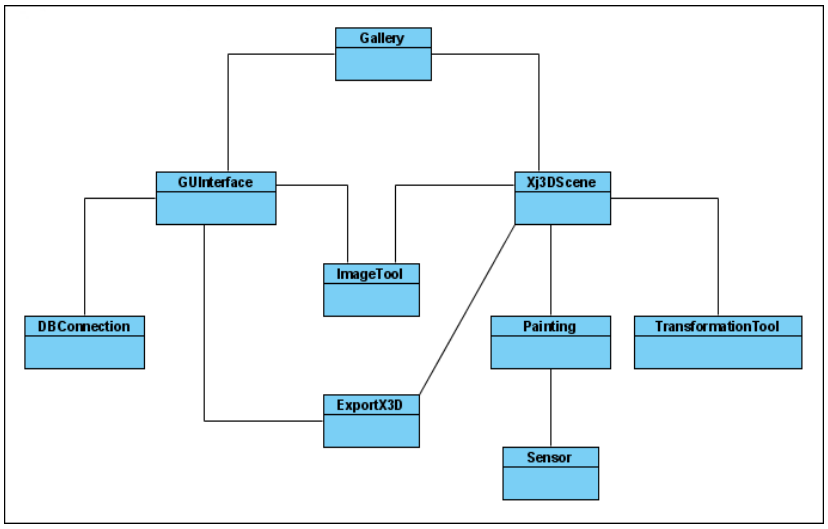

Figure 4: Virtual Exhibition Builder Simplified Class Diagram.

5.1 Application's Classes and Responsibilities

The simplified class diagram can be seen in Fig. 4, from which we will highlight the key components and their function.

The Gallery class holds the instances of the two main classes: GUInterface and Xj3DScene.

GUInterface is responsible for all the menus. It holds an instance of the class responsible for the database connection, which it uses for painting access and filtering. It also uses the ImageTool class to generate thumbnails of the paintings in real time.

Xj3DScene is responsible for the display and interaction with the X3D model. It keeps a collection of Painting's instances, which represent the paintings added to the scene. Each one of these instances has a Sensor associated, with information about the painting's sensor. The TransformationTool class is responsible for all calculations involved in 3D transformations, like placing and moving paintings. ExportX3D is responsible for all the X3D code, also saving it to a new X3D file when the user chooses to save his work.

5.2 X3D Code Handling

The X3D code is kept in three separate containers, as seen in Fig. 5.

The first container, PRE, holds all the X3D code until the end of scene tag.

The collection of containers in the middle holds the shapes of all the paintings added to the scene. Each one of these containers is divided into three sub containers. The B container has the coordinates of the shape, pertaining to the Coordinate X3D tag, A and C contain, respectively, the code before and after the shape's coordinates.

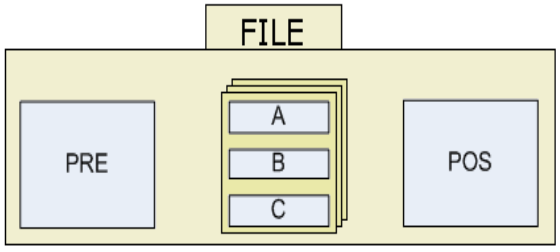

Figure 5: X3D File Contents.

This allows for the updating of the paintings' coordinates when needed.

The last container, POS holds the end of scene tag and the remaining code.

When the user saves his work the file is written, starting with PRE, then all the containers in the collection, and finally POS is written.

5.3 Handling the Scene with XJ3D

When the application starts all the sensors in the X3D file are loaded into a collection of sensors and the coordinates of each sensor are also stored. A listener is created and all the sensors are added to it. This will allow the application to catch the activation of any sensor. When a sensor is activated, its coordinates are used to place the currently selected painting on the corresponding surface. The scene is updated, but it is not possible to retrieve the new X3D code from the scene. Therefore we generate the X3D code for the new painting, by passing its information to the ExportX3D class.

The newly placed paintings must also have sensors, so that they can be selected to adjust their position. Each Painting instance holds an instance of the Sensor class, with all the information about the sensor. When a painting is selected, the Image Tool menu becomes available. A new shape is created between the painting and the wall to give feedback to the user about the selection. If the user chooses to delete the painting, its node is removed from the scene and the ExportX3D instance is informed of its deletion, so it can update its X3D code accordingly. If the user chooses to move the painting several steps are taken:

- Its parameters and desired movement are passed into TransformationTool's instance to calculate the new coordinates.
- The scene is changed to reflect the new position, by altering the shape's Coordinates node. The coordinates for the visual selection indicator are also updated.
- The painting's sensor is updated with the new coordinates.
- The painting is marked as changed, signaling that it has moved from its original position.

ExportX3D's instance code is not updated at this time. When the user chooses to save, all the new paintings marked as changed will be updated in ExportX3D's instance.

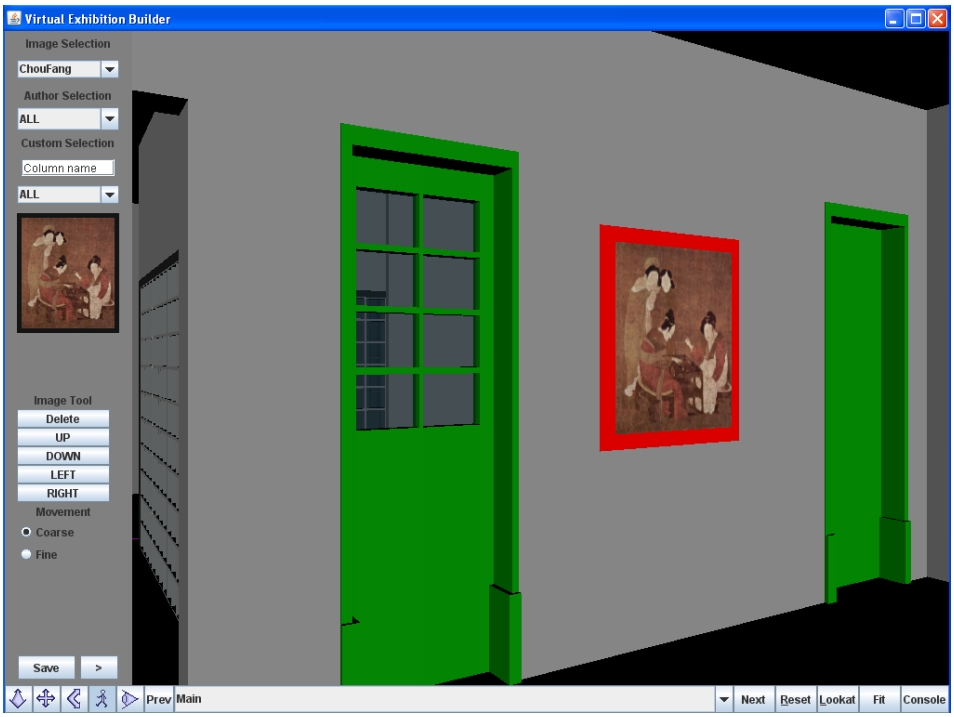

Figure 6: Virtual Exhibition Builder Screenshot.

5.4 Using the Application

We can see a screen capture of the application at Fig. 6. The bottom menu is the Xj3D specific menu. It is used to change the view mode and toggle among viewpoints.

The menu on the left is the application menu. At the top of the menu we can see the interface to the database of paintings, where we can filter and choose paintings. There is a custom filter box which allows for accommodation of database tables with more columns then the ones originally specified. Below the painting's thumbnail we have the image tool menu. This is a context sensitive menu that is only visible when there is a painting selected in the scene. Here we can delete a painting from the scene and move a painting through the surface on which it is placed. The radio buttons control the coarseness of the movement. The final row of buttons is composed by the Save button, which saves the work done to a new X3D file, and a button to shift the menu to the other side of the screen, for ease of use for users with both dexterities.

The remainder of the screen is occupied by the X3D scene. Here the user can navigate throughout the scene and chose a surface with the mouse. The currently selected painting on the menu will then be placed in the center of the surface. Only the surfaces chosen with the Space Picker application are able to receive paintings.

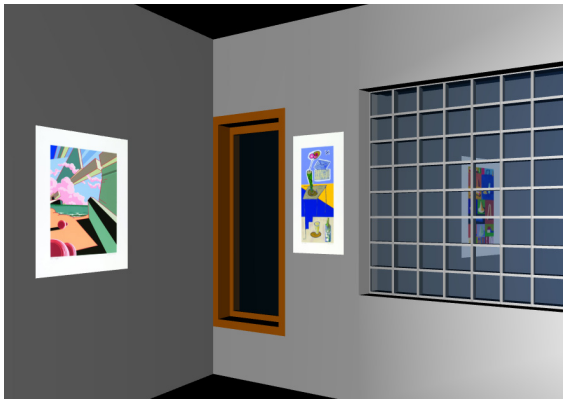

Figure 7: IAC's Exhibition Sample.

6 PRESENT AND FUTURE WORK

We are using the applications to help the Azorean Cultural Institute (IAC) to build virtual exhibitions with their art collection. An exhibition will be hosted on the virtual model of its headquarters. We can see a sample of this work at Fig. 7, a screen capture of a resulting X3D scene.

There has been some interest in the application by art students to spread their work through the Internet.

One interesting goal would be the ability to add 3D models to the scene too.

The application has not been tested with X3D models with textures, and probably would require some tweaking with the X3D code handling routines to work properly.

In relation to the previous issue, there is also some interest in developing an Xj3D based application for texture placement in X3D models.

Joining the two existing applications with a new one, for managing the painting database, would make this tool more complete and usable.

7 CONCLUSIONS

The Virtual Art Gallery tool makes the task of organizing a virtual art show in an X3D model a matter of a few clicks. Use of the applications is very intuitive and simple. In our tests with the Virtual Exhibition Builder application we managed to create exhibitions with ease and consistency, and in a matter of a few minutes.

The use of free software in the different facets of the tool, interface, application logic, 3D visualization and manipulation, and database access, is, in our view, an important attribute of the tool, which we hope will lead to its broader acceptance.

ACKNOWLEDGEMENTS

We would like to publicly express our appreciation to IAC for the motivation in the project goals and all the information, blueprints and painting scans, made available to us.

REFERENCES

httpLouvre, 2007. http://www.louvre.fr/llv/musee/visite_virtuelle.jsp?bmLocale=en

httpQTVR, 2007. http://www.apple.com/quicktime/technologies/qtvr/

httpWeb3d, 2007. http://www.web3d.org/

Wojciechowski, R; Walczak, K; White, M; Cellary, W., 2004. Building Virtual and Augmented Reality Museum Exhibitions. In Proceedings of the ninth international conference on 3D Web technology, pp 135-144

Hrk, S., 2001. Virtual Art Gallery. In Central European Seminar on Computer Graphics.

Bimber, O.; Frohlich, B; Schmalstieg, D.; Encarnação, L., 2001. The Virtual Showcase. In IEEE Computer Graphics and Applications, volume 21, nr. 6, pp 48-55

httpArco, 2007. http://www.arco-web.org/index.html

Walczak, K.; Cellary, W., 2003. X-VRML for Advanced Virtual Reality Applications. In IEEE Computer; volume 36, nr 3, pp 89-92

Walczak, K., Wojciechowski, R., 2005. Dynamic Creation of Interactive Mixed Reality Presentations. Proceedings of the ACM Symposium on Virtual Reality Software and Technology, pp 167-176