VEX-CMS: A tool to design virtual exhibitions and walkthroughs that integrates automatic camera control capabilities

Luca Chittaro, Lucio Ieronutti, Roberto Ranon

HCI Lab, University of Udine, via delle Scienze, 206,
33100 Udine, Italy
http://hcilab.uniud.it

Abstract. This paper presents VEX-CMS, a tool to build 3D virtual museums and exhibitions, architectural walkthroughs and, more generally, applications where 3D and 2D content is presented to the user inside a 3D Virtual Environment. In this paper, we concentrate on the task of designing tours of the virtual environment to be followed by visitors, either through manual or automatic navigation, and show how a recent automatic camera control approach [3], coupled with path planning, can be exploited to accomplish the task with minimal knowledge and manual effort on the curator's side.

Keywords: 3D virtual museums, automatic camera control, 3D authoring tools.

1 Introduction

A number of graphics applications, such as 3D virtual museums and exhibitions, or architectural walkthroughs, are often built around the idea of having the user enter a 3D Virtual Environment (hereinafter, VE) and walk around (or follow pre-computed paths), see objects and places in the environment and possibly get information on them.

Although this sort of applications has been built since the 80s, they are still difficult to produce without the help of 3D graphics and computer professionals. A number of projects has focused on this issue, concentrating on problems such as facilitating the acquisition of 3D models of existing artifacts [1] or organizing collections of artifacts into digital libraries with metadata [2], from which 3D virtual exhibitions can then be assembled.

In this paper, we take a different approach and focus on museum or exhibition curators who, starting from available 3D models of a building and objects of interest, want to build a 3D virtual exhibition (or architects who want to build an interactive walkthrough of a virtual building). More specifically, we concentrate on the task of designing tours of the VE to be followed by a visitor, either with manual or automatic navigation, and show how a recent automatic camera control approach [3] can be exploited to accomplish the task with minimal knowledge and manual effort on the curator's side.

The paper is organized as follows. Section 2 reviews related work on tools for the construction of virtual exhibitions. Section 3 briefly presents the application we have developed, called *VEX-CMS* (*Virtual EXhibition Content Management System*). Section 4 concentrates on tour and camera control aspects. Section 5 briefly presents examples of applications built with VEX-CMS. Finally, Section 6 concludes the paper and outlines future work.

2 Related Work

Virtual museums and exhibitions have been studied in different areas (e.g. human-computer interaction, 3D graphics and virtual reality, database systems) and research has been carried out on many aspects involved in their construction. For example, Patel et al. [1] reports the main outcomes of the European Project ARCO in developing a complete solution for digitizing artworks, storing them into multimedia databases together with relevant metadata [2], and presenting them in the context of 3D or augmented-reality exhibitions. However, there has not been much work on the specific aspect of making the design of the interactive 3D exhibition easy for non technical users such as curators and architects. One possibility that is often suggested is to exploit game engine tools. However, as noted by one of the proponents of this approach [4], programmers are still needed, and work is required to remove game-specific features and ultimately adapt the logic of the tool to a different purpose.

Some research for simplifying the creation of VEs for non technical users has been carried out considering the special case of children. For example, Kids Movie Creator (KMC) [5] is a tool that allows young children (aged 7 to 12) to create interactive VEs. KMC introduces many solutions to simplify interaction both at the conceptual and practical levels, from adopting an avatar-based interaction (i.e., the author of the VE controls an avatar that walks through the environment) to limiting modeling capabilities and object arrangement with discrete steps.

Alice [6] is a 3D programming environment designed for undergraduates with no 3D graphics and programming experience that allows to build VEs and program their behavior. Our work shares some of the assumptions and goals behind Alice, e.g. the fact that VE creation could be interesting for a much broader audience that will not necessarily have a mathematical and programming background, the importance of designing a VE creation tool with a target audience in mind, and providing users with simple methods to specify VE logic.

Assisted or semi-automatic camera control has been experimented in the context of virtual museums, but typically for helping the visitor find its way and easily navigate through the museum. For example, the research presented in [7] develops an intelligent camera control system and applies it to the task of navigating a virtual museum, using constraints to specify camera requirements. We also exploit a constraint-based approach but focus on the largely unexplored topic of using automatic camera control in the authoring process of a VE. The approach proposed by Elmqvist et al. [8] automatically generates a tour of a VE that includes all landmarks; in this way, however, the curator has no control on how the visitor will experience the

virtual exhibition, and so this approach might not be well suited to virtual exhibitions or architectural walkthroughs.

To introduce automatic camera control in the authoring process, Bares et al. [9] adopt a visual interface to specify constraints and compute static cameras, with the goal of simplifying the process of camera placement for the non-technical user. While this kind of interface could be useful in a virtual exhibition context, it would still require the curator to concentrate on aspects such as the position, size and angle of objects in the derived camera. In our approach, we instead choose to provide a simpler and higher-level interface to the curator, only focusing on which objects should be shown by a camera and their relative importance.

3 The Virtual Exhibition Content Management System

VEX-CMS organizes the design of a virtual exhibition in two phases: arrangement of objects of interest in the VE, and design of virtual tours of the VE. In the following, we use the terms *curator* to indicate the creator of the VE, and *visitor* to indicate the user of the VE.

The main interface (see Figure 1) of VEX-CMS is organized in two parts: (i) a first-person view of the VE, where the curator's avatar can walk (or fly) and arrange objects in space, (ii) a series of tabbed panes, that describe the *objects of interest* (*items*), *views of interest* (*stops*), and *tours* that make up the virtual exhibition (the last two features will be described in detail in Section 4). The system is not meant for object modeling purposes, in the sense of creating or modifying 3D models, since that is typically outside the abilities of a curator and can be done better with existing 3D modeling tools such as Maya, 3DS or Blender. To build a virtual exhibition with VEX-CMS, a curator needs an existing 3D model of the VE and 3D models of all the objects she wants to arrange in the VE (by means of translation, rotation and scale operations).

The first-person view of the VE can be augmented with an overlaid interactive map in a layout that is significantly more efficient than using just the first-person perspective, as we determined in an experiment with non-technical users [10]. Moreover, we added the possibility of introducing *snap-to* constraints when positioning objects, by specifying the side of the object (e.g. its bottom) that should stay attached to adjacent geometry (e.g., the floor or an artwork support). In this way, whenever the object is not attached to anything on the specified side, it will "fall" in a direction perpendicular to that side (e.g. down, if bottom is used as the chosen side) until it finds a geometry on which to stand.

The idea we followed in designing the application was to closely reproduce the experience of designing a real exhibition, where the curator visits the actual exhibition space and decides where to position objects and explanations. However, one important part of real exhibitions, i.e. lighting, cannot be directly controlled inside the application, but has to be incorporated into the 3D modeling process.

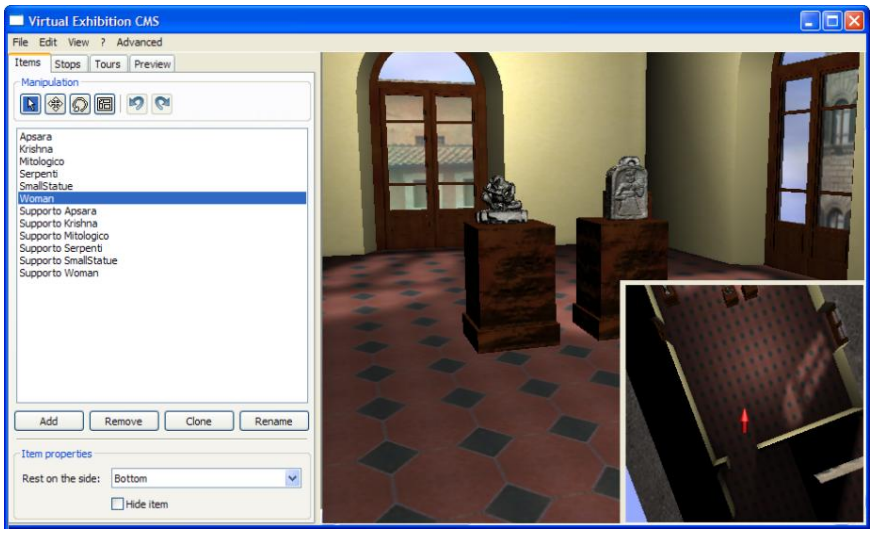

Figure 1. The main interface of VEX-CMS. On the right, a first person perspective of the VE (with the overlaid interactive map) where the curator can navigate (by walking or flying) and arrange objects; on the left, the list of currently inserted objects, and buttons to manage and arrange objects.

4 Designing Virtual Tours

In our approach, virtual tours are based on the notion of *View of Interest* (VOI), which extends the more typical notion of *Point of Interest*. A VOI is a point in the VE to which we associate: (i) a camera directed at something interesting in the context of the exhibition (e.g., to show one or more artworks), and (ii) related information, represented in HTML files that can be displayed on a 2D overlay or applied as a texture to some geometry in the VE. Therefore, the concept of VOI encapsulates both a point of interest in the VE and an interesting view direction from that point, or, in other words, a set of objects that can be appreciated from that point of view.

We support the definition of VOIs in two ways: manual and automatic. In the first case, the curator simply defines a VOI by "taking a picture" from her current viewpoint. In this way, defining a VOI requires one to navigate the environment until the desired view is reached. In the automatic case, the curator chooses which objects need to be included in the VOI and their relative importance (see Figure 2), and then a VOI which best satisfies these requirements is computed. The VOI can then be used as it is, or refined with manual navigation (e.g. slightly changing the position or orientation to get a better view). In this last case, the system will store the manual VOI instead of the automatically computed one.

VOIs can be visually connected to form one or more tours of the VE (see Figure 3). Tours are directed graphs of VOIs, with conditions for activating transitions (from a VOI to another) and information on how the transition will be performed (manual

navigation with navigation aid, automatic navigation, or teleporting). In the following, we describe these features in more detail.

4.1 Automatic computation of VOIs

Automatic computation of VOIs is performed by exploiting the technique described in [3], which uses a declarative approach to camera composition problems. The user (or the application, in this case) sets a number of requirements for the desired camera, and then the position and orientation of the camera that best satisfies those requirements is computed through a constrained optimization process.

More specifically, the requirements that can be expressed are in the form of properties that can either directly bind camera parameters (e.g. to avoid upside-down cameras), or express geometric requirements on camera parameters with respect to objects in the scene (e.g. to look at a certain side of an object), or express requirements on the image the camera will generate, e.g. requiring a certain object to be unoccluded and of a certain size. The constrained optimization process then treats some of the requirements as constraints that cut the 6DOF (camera position and orientation) search space by defining so-called *feasible volumes* (i.e., portions of the scene in which the camera can be positioned to satisfy the constraints), and search inside the feasible volumes is carried out using a global optimization strategy called *Particle Swarm Optimization* (*PSO*) [11].

To use automatic computation of a VOI, the curator chooses a set of objects to be included in the VOI, possibly filtering the list of all available objects by selecting one of the rooms in the VE (this also constrains the search space considered by PSO, and helps focusing the optimization process). The curator can also order the objects by importance (see Figure 2), and the application uses that information to generate a set of camera control requirements that are provided as input to the automatic computation.

More specifically, for each object *obj* that has to be included in the VOI, the following requirements are generated (see [3] for more details on the semantics of the requirements):

objInFOV (obj, 1.0, weight)
objHAngle(obj, 0, 40, weight)
objOcclusion(obj, 0, weight)
objProjSize(obj, size, weight)

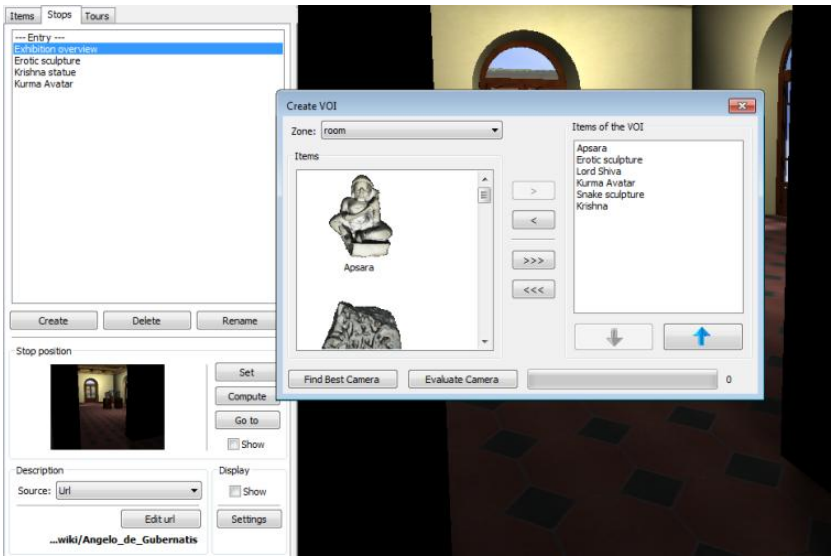

Figure 2. The interface for defining a VOI. In the bottom left part, one can see a preview of the VOI, and buttons to set a VOI from the current viewpoint, compute the VOI automatically, or go to a VOI. The second option opens the dialog window displayed in the center of the screen, which allows the curator to choose the objects to be included in the VOI (left part of the dialog window) and set their relative order of importance (right part of the dialog window).

The first requirement (*objInFOV*) states that the object should be entirely included into the camera view volume, i.e. uncut in the image produced by the VOI camera (the 1.0 value is the required percentage of the object that must be in the camera view volume). The second requirement (*ObjHAngle*) states that the object should be seen from its front direction, with 40 degrees of tolerated deviation (the 0 value is the deviation with respect to the object front vector). The third requirement (*ObjOcclusion*) states that the object should not be occluded by other objects in the image produced by the camera (the 0 value is the required percentage of occlusion). The final requirement (*objProjSize*) states that the projected size of the object in the image produced by the camera should be equal to *size*, which is a function of the number of objects that are included in the VOI (i.e. *size* is proportional to *1/(number of objects)*, considering the image having an area equal to 1). Finally, all statements include a weight parameter, which defines how much a requirement counts in the objective function that is maximized with PSO (see [3] for more details) and is greater for more important objects.

Moreover, we add a number of requirements directly related to the camera, i.e. *camBindY (avatarHeight, avatarHeight)*, *camBindRoll(0,0)*, which set minimum and maximum values for the camera height (both set at the height of the visitor's avatar) and roll (which is then set as zero).

A possible issue with this approach might occur when the curator includes a set of objects which produce an over-constrained situation, i.e. a set of requirements that cannot be all satisfied by a single camera. The requirements we generate are all used just in the optimization process, with the exception of *objHAngle*, which is used also

as a constraint to cut the search space. This means that, except for *objHAngle* properties, the computation will return, even in over-constrained situations, a result which tries to satisfy as much requirements as possible, giving precedence to those with higher weight (i.e. to the most important objects). If the problem is over-constrained because of *objHAngle* properties (i.e. the search space becomes empty), we remove those constraints in the search space cutting process, and repeat the computation just using them in the optimization function, with the only drawback of taking more time to compute the result.

When the computation of the VOI ends, the result is shown to the curator together with a bar that graphically shows how much the requirements have been satisfied. The curator can accept the result (and possibly refine it manually) or edit the requirements to fine-tune them or add other requirements (e.g. framing).

4.2 Connecting VOIs to Create Tours

Once a set of VOIs has been defined, the curator can connect them to create tours that a visitor can follow. Tours are displayed in the interface as graphs where VOIs are nodes, and possible transitions are represented as directed arcs that connect them (see Figure 3). The curator graphically lays out VOIs on a plane, and draws transitions as lines between them. Moreover, she can define multiple tours, which will then be proposed as alternatives when the visitor enters the VE.

From a formal point of view, the creation of a tour defines a finite state automaton, where states correspond to VOIs, and therefore, at each moment of a visit to the VE, one VOI will be the active one. Transitions between states are defined by choosing a condition for starting the transition and specifying how the transition has to be executed.

Starting conditions include events such as time elapsed since the starting VOI became active or actions done by the visitor, such as clicking on a certain item or moving away from the VOI. The transition can then be executed in three possible ways: by teleporting the visitor, by automatically navigating her avatar to the destination VOI (in this case, an animation will lead the visitor to the VOI, simulating a walk), or by drawing a path to the destination VOI (in this case, the visitor has to manually navigate the VE following the visual help provided by the drawn path).

This approach for specifying tours allows one to define different types of visits: from completely assisted ones (the visitor does not need to perform any navigation activity and is guided through the VE as if she were inside a virtual vehicle) to more interactive ones (the visitor can actively navigate the VE possibly with the help of visual paths). To define completely free visits (allowing the visitor to choose any order in which objects are seen), there is also the possibility of simply specifying enter and exit conditions for a VOI, without indicating a transition. The enter condition specifies the cases in which the automaton switches from a default state to a specific VOI state (e.g. when the visitor is sufficiently near to the VOI), while the exit condition handles the opposite situation (e.g. when a certain amount of time has elapsed since entering the VOI, we can go back to the default state).

For path planning, we adopt a cell decomposition approach. In particular, using an automatic approach similar to the one described in [12], for each room of the VE

(which is defined by bounding volumes) VEX-CMS derives a raster image in which each pixel corresponds to an area of the virtual room and the color of the pixel indicates whether the corresponding area contains a geometry (i.e., cannot be navigated) or not.

The path planner uses such images as well as connectivity information between rooms for building an adjacency graph, i.e. an undirected graph in which nodes corresponds to navigable areas while edges indicate if two areas are adjacent. Then, we compute collision-free paths in two steps, as in [13]. First, we derive the sequence of nodes that connect the nodes representing the start and end position by using Dijkstra's algorithm. Second, we compute a collision-free path connecting the centers of the cells corresponding to the sequence of nodes obtained from the previous phase.

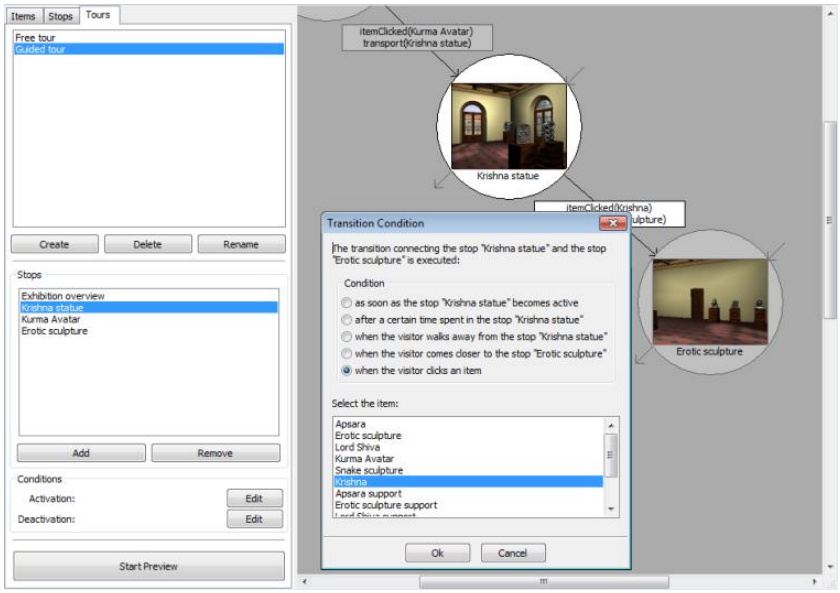

Figure 3. The interface for defining a tour. VOIs are shown as icons in the right part of the interface, they can be arranged spatially and then connected by drawing arcs among them.

5 Example Application

VEX-CMS is currently being used by a team of Indian art historians and experts to rebuild an Indian museum that was hosted in Florence, Italy at the beginning of the 19th century. 3D reconstructions of artworks have been obtained through laser scanning, while a 3D model of the original environment has been built from pictures of the original exposition.

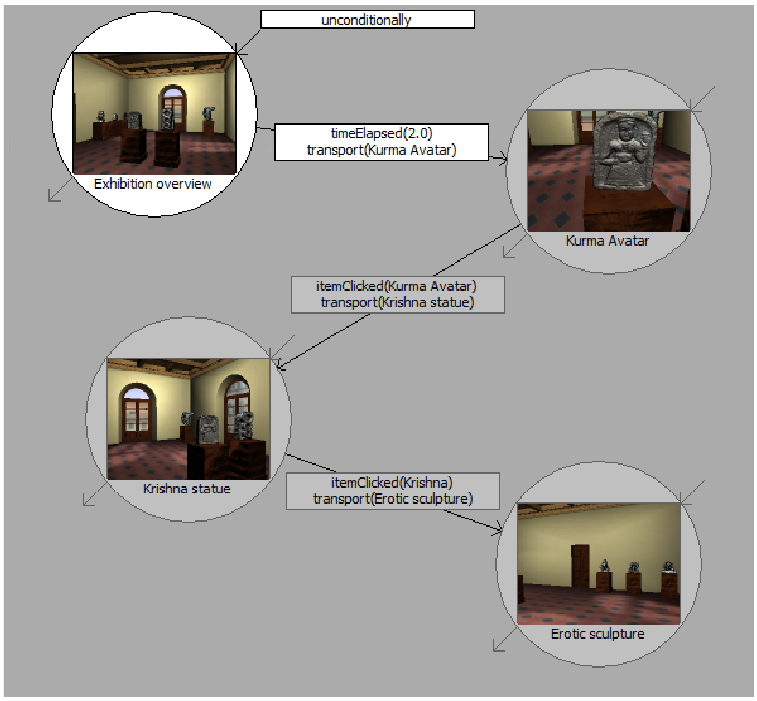

Figure 4. Automated tour for a room of the Indian exhibition.

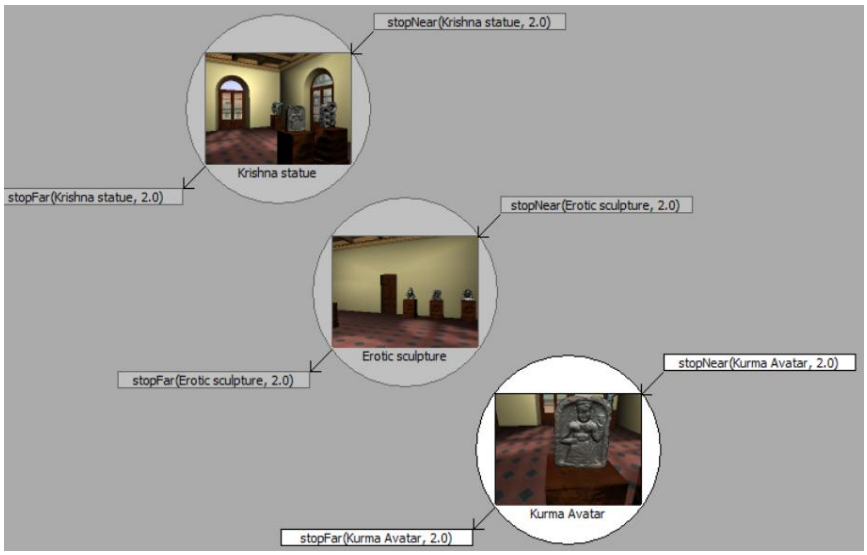

Figure 5. Free tour for a room of the Indian exhibition.

Figure 4 shows a fully automatic tour that was designed for one of the rooms of the virtual exhibition. As soon as the visitor enters the room, the automaton goes into the first VOI state (top left VOI in Figure 4), which corresponds to an overview of the room (shown in Figure 6). After a few seconds, the visitor is transported to the first artwork, and associated information is overlaid on the screen. After clicking an artwork, the visitor is transported to the following one in the tour.

Figure 5 shows a free tour that was designed using the same VOIs. In this case, the visitor is free to navigate and visit artworks in any order, and a VOI is activated (or deactivated) every time the visitor is sufficiently near to (far from) it.

Finally, Figure 6 shows an example of a camera computed from requirements for all the six artworks in the room, which is an over-constrained situation (for example, not all objects can be seen from their front side).

Figure 6. An automatically computed camera showing all the artworks in a room of the virtual exhibition.

6 Discussion and Conclusions

In this paper, we have presented an authoring tool for 3D virtual exhibitions and walkthroughs that uses intelligent camera control and path planning to make the design of virtual tours easier, and allows a curator to manage visitor's navigation in a new way, insteadof simply relying on pre-computed paths or navigation aids such as

arrows or signs. This flexibility opens up the possibility of mimicking and supporting typical visiting styles of exhibitions that have been observed in real museums. For example, Veron and Levasseur [14] have conducted ethnographic studies of museum visitors and have identified four categories of visitors, called *ant*, *fish*, *grasshopper* and *butterfly*. The ant visitor spends quite a long time to observe all exhibits, stops frequently and usually moves close to walls and exhibits. The fish visitor moves preferably in the center of rooms, and does not look at details of exhibits, making just a few or no stops. The grasshopper visitor sees only some exhibits, each for quite a long time, choosing them on the basis of personal interests and pre-existing knowledge about the contents of the exhibition. The butterfly visitor changes frequently the direction of visit, and sees almost all the exhibits, stopping frequently, but times vary for each exhibit. While the last two styles are only suited to free visits, the first two (ant and fish) can be respectively implemented by creating a tour composed by a VOI for each exhibit (the ant style) or less VOIs, but including more exhibits located in the same room (the fish style).

A limitation of our current approach is in the kind of exhibitions that can be designed. In particular, contemporary art exhibitions often include non-physical works (videos, interactive computer-based visualizations, …), or exploit complex interactions of dynamic lights with the exhibitions space (e.g. lasers). However, while some of these features could be quite easily incorporated (e.g. videos), others are generally difficult to reproduce in a VE.

Directions for future work include experimenting with alternative ways of specifying VOIs from high-level requests (e.g. interfaces to easily specify framing properties) as well as providing more sophisticated results from the automatic VOI computation. For example, to handle over-constrained situations, instead of producing a single VOI, we could generate an establishing shot followed by several shots of smaller groups of items.

Another important direction for expanding the possibilities in the authoring phase is the incorporation of semantics into the artworks, as also stressed by the 3D-COFORM project [15]. This would allow the automatic camera control process (and therefore the curator) to target also specific parts or features of artworks.

Finally, we are currently evaluating the usability and effectiveness of VEX-CMS with several curators..

Acknowledgements

The authors acknowledge the financial support of the Italian Ministry of Education, University and Research (MIUR) within the FIRB project number RBIN04M8S8.

References

1. Patel, M., White, M., Walczak, K., Sayd, P.: Digitisation to Presentation - Building Virtual Museum Exhibitions. In: Proceedings of International Conference on Vision, Video and Graphics, pp. 189-196 (2003)

2. Mourkoussis, N., White, M., Patel, M., Chmielewski, J., Walczak, K.: AMS: metadata for cultural exhibitions using virtual reality. In: Proceedings of International Conference on Dublin Core and Metadata Applications, Dublin Core Metadata Initiative, pp. 1-10 (2003)
3. Burelli, P., Di Gaspero, L., Ermetici, E., Ranon, R.: Virtual Camera Composition with Particle Swarm Optimization. In: Proceedings of SG 2008: 8th International Symposium on Smart Graphics, LNCS, Vol. 5166, pp. 130-141. Springer-Verlag, Berlin Heidelberg (2008)
4. Lepouras, G., Vassilakis, C.: Virtual museums for all: employing game technology for edutainment. Virtual Reality, Vol. 8(2), 96-106 (2004)
5. Wang, T., Li, X., Shi, J.: An Avatar-Based Approach to 3D User Interface Design for Children. In: Proceeding of the Symposium on 3D User Interfaces, pp. 155-162. IEEE Computer Society Press, Los Alamitos (2007)
6. Conwaym, M., Audia, S., Burnette, T., Cosgrove, D., Christiansen, K., et al.: Alice: lessons learned from building a 3D system for novices. In: Proceedings of the Conference on Human Factors in Computing Systems, pp. 486-493. ACM Press, New York (2000)
7. Drucker, S.M., Zeltzer, D.: Intelligent camera control in a virtual environment. In: Proceedings of Graphics Interface, pp. 190-199 (1994)
8. Elmqvist, N., Tudoreanu, M., Tsigas, P.: Tour Generation for Exploration of 3D Virtual Environments. In: Proceedings of VRST-2007: 14th ACM Symposium on Virtual Reality Software and Technology, pp. 207-210 (2007)
9. Bares, W., McDermott, S., Boudreaux, C., and Thainimit, S.: Virtual 3D camera composition from frame constraints. In: Proceedings of MULTIMEDIA 2000: 8th ACM international Conference on Multimedia , pp. 177-186 (2000)
10. Chittaro, L., Ranon, R., Ieronutti, L.: 3D Object Arrangement for Novice Users: the Effectiveness of Combining a First-Person and a Map View. In: Proceedings of VRST-2009: 16th ACM Symposium on Virtual Reality Software & Technology, pp. 171-178. ACM Press, New York (2009)
11. Eberhart, R.C., Kennedy, J.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, Vol. 4, pp. 1942–1948. IEEE Press, Piscataway, New Jersey (1995)
12. Ieronutti, L., Ranon, R., Chittaro, L.: Automatic Derivation of Electronic Maps from X3D/VRML Worlds. In: Proceedings of Web3D 2004: 9th International Conference on 3D Web Technology, pp. 61-70. ACM Press, New York (2004)
13. Kuffner, J.: Autonomous agents for real-time animation. Ph.D dissertation, Stanford University (1999)
14. Veron, E., Levasseur, M. : Ethnographie de l'Exposition. Bibliothque publique d'Information, Centre Georges Pompidou, Paris(1983)
15. Arnold, D.: 3D-COFORM: Tools and Expertise for 3D Collection Formation. In: Proceedings of EVA2009 Florence, pp. 94-99 (2009)