A Generic Model for Embedding Users' Physical Workspaces into Multi-Scale Collaborative Virtual Environments

Cédric Fleury*
INSA, IRISA UMR CNRS 6074, Rennes, France

Alain Chauffaut †
INRIA Rennes Bretagne-Atlantique, France

Thierry Duval ‡
Université de Rennes 1, IRISA UMR CNRS 6074, Rennes, France

Valérie Gouranton §
INSA, IRISA UMR CNRS 6074, Rennes, France

Bruno Arnaldi ¶
INSA, IRISA UMR CNRS 6074, Rennes, France

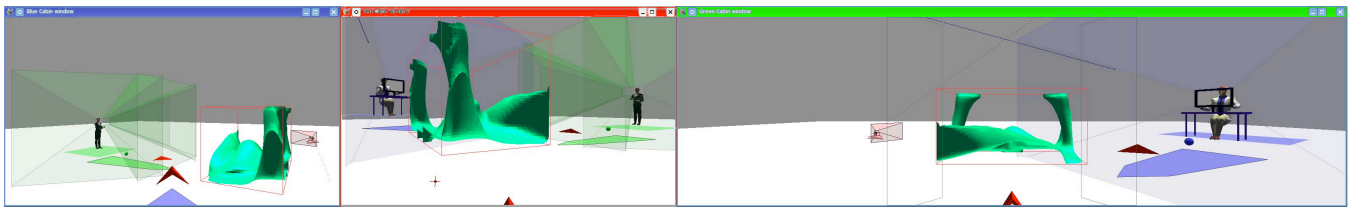

Figure 1: Three users with their own IIVC navigating and interacting within a Multi-Scale Collaborative Virtual Environment: each user can perceive other users' physical workspaces

Abstract

Most Virtual Reality (VR) systems must consider the users' physical environment to immerse these users in a virtual world and to make them aware of their interaction capabilities. However, no consensus has been found in the existing VR systems to embed the real environment into the virtual one: each system meets its particular requirements according to the devices and interaction techniques used. This paper proposes a generic model that enables VR developers to embed the users' physical environment into the Virtual Environment (VE) when designing new applications, especially collaborative ones. The real environment we consider is a multi-sensory space that we propose to represent by a structured hierarchy of 3D workspaces describing the features of the users' physical environment (visual, sound, interaction or motion workspaces). A set of operators enables developers to control these workspaces in order to provide interactive functionalities to end-users. Our model makes it possible to maintain a co-location between the physical workspaces and their representation in the VE. As the virtual world is often larger than these physical workspaces, workspace integration must be maintained even if users navigate or change their scale in the virtual world. Our model is also a way to carry these workspaces in the virtual world if required. It is implemented as a set of reusable modules and used to design and implement multi-scale Collaborative Virtual Environments (msCVE). We also discuss how three "state of the art" VR techniques could be designed using our model.

Index Terms: H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces—Theory and methods; H.5.1 [Information Interfaces and Presentation (e.g., HCI)]: Multimedia Information Systems—Artificial, augmented, and virtual realities ; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques, Device independence

*e-mail: cedric.fleury@irisa.fr
†e-mail: alain.chauffaut@inria.fr
‡e-mail: thierry.duval@irisa.fr
§e-mail: valerie.gouranton@irisa.fr
¶e-mail: bruno.arnaldi@irisa.fr

20th International Conference on Artificial Reality and Telexistence (ICAT2010), 1-3 December 2010, Adelaide, Australia
ISBN: 978-4-904490-03-7 C3450 ©2010 VRSJ

1 Introduction

Nowadays Virtual Reality (VR) applications are used in many different fields such as industrial design, scientific data visualization and training, etc. Each kind of application requires specific interaction techniques and can be used on various physical environments from full immersive devices to "non-immersive" devices. To improve user presence, VR developers must consider the users' physical environment when designing one of these applications. This makes it possible to match the real world with the virtual world (co-location), to increase the users' sensation of presence, and to inform users about their interaction capabilities (these capabilities are often limited by the size of the users' physical workspaces).

As an example, Figure 1 presents the viewpoints of three users collaborating in a multi-scale Collaborative Virtual Environments (msCVE): each user carries his interaction tools (instances of 2D Pointers/3D Rays [8] dedicated to collaborative interaction) within his Immersive Interactive Virtual Cabin (IIVC). This area, where the user can move and interact with co-located virtual objects, is represented as a colored virtual flying carpet. The IIVC also embeds a representation of its user's field of view. So each user can perceive the interactions limitations of the other users: he can see what they can reach with their virtual hand inside their IIVC, or what they can reach with a virtual ray inside their field of view.

Despite the fact that several existing VR applications take into account the features of the users' physical environment, such as for natural walking applications [6] (see Figure 3), no consensus has been reached for these VR systems on how to embed this real environment into the virtual one. Each system models the real world in a particular way according to its requirements. We want to fill in the lack of standard metaphors for 3DUI to simplify Virtual Environment Design and Implementation [26]. We are interested in modeling the system, not only through device abstraction, but also by modeling the physical users' workspaces and the relationship

between these workspaces.

So we propose the Immersive Interactive Virtual Cabin as a generic model that enables VR developers to embed the users' physical environment into the Virtual Environment (VE) when designing or deploying new applications. This high-level model depicts the relationships between the real and the virtual world whatever the physical devices used or the room physical configuration.

Our model deals with multi-scale collaborative virtual environments (msCVE as described in [27]). This kind of virtual environment is more and more used to enable remote experts to work together on multi-scale structures such as scientific data or chemical molecules. Our model solves issues induced by collaborative sessions with remote users who interact from different physical environments: it enables users to perform a more effective collaboration by providing them a better understanding of the others' interaction capabilities, as it integrates users' physical workspaces and interaction tools in the virtual environment (see Figure 2).

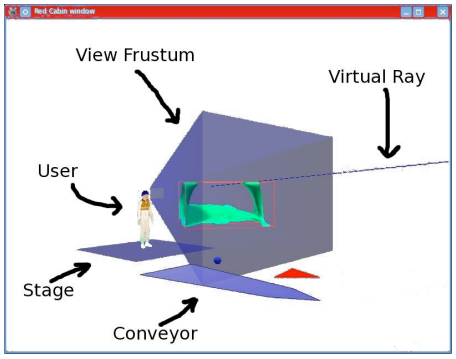

Figure 2: One user within his IIVC viewed by another user: we can see him, his conveyor, his stage, his view frustum and his virtual ray.

As user presence is a multi-sensory perception of the virtual world, we propose to model the users' physical environment as a structured hierarchy of sensory workspaces, such as motion, visual, sound, interaction or haptic workspaces. For example, a motion workspace is the area where a user can move his body. A visual workspace corresponds to a display device. A sound workspace represents the area in which a user perceives sound. An interaction workspace is the area where a user can interact. A haptic workspace is the area where a user can interact and have feedback when he uses a haptic device. The structured hierarchy of workspaces depicts the real-world spatial relationships between these workspaces. Our model also defines a set of operators to control each workspace, which enables VR developers to provide interactive functionalities to end-users.

This paper is organized as follows. Section 2 presents related work about the necessity of embedding the users' physical workspaces and about VR system design. Section 3 presents an overview of the IIVC concept, then section 4 describes the IIVC model, its structure and its operators. Section 5 describes the functionalities it offers to end-users for interaction and collaboration, and how they can be implemented by VR developers. Section 6 illustrates how this model has been instantiated to design and implement Collaborative Virtual Environments (CVE), and how it could be used to design other existing VR techniques. Finally, section 7 concludes the paper and section 8 gives some directions for future research on this topic.

2 Related Work

Previous work about user interaction aims to provide users with a truly immersive experience. To achieve this, an increasing amount of this work has to embed the users' physical workspaces into the VE to consider the users' interaction capabilities (see part 2.1). Although they often propose interesting hierachical data-structures, device abstractions and customization at run-time to adapt to physical devices, none of the existing software models take into account these physical workspaces in the VE when designing a new VR application (see part 2.2).

2.1 Embedding the Physical Workspaces into the VE

Some previous work has embedded the user's motion workspace into the virtual environment. This offers the user an intuitive way to navigate by moving his own body. It also makes it possible to manage problems induced by the fact that the virtual world is often larger than this workspace. For example, the 3DM graphical modeler [5] enables a user to move on a "magic carpet" which represents the boundaries of the tracking area. The user uses Head Mounted Display (HMD), so he can perform real movements on the "magic carpet" to intuitively perform interactions. For long-distance navigation, he can also drive the "magic carpet" into the virtual world with a specific tool. The user reaches interaction tools through a 3D menu, which can be put on the "magic carpet" in order to travel with it. For natural walking in virtual worlds with a restricted workspace, the "Magic Barrier Tape" [6] displays the boundaries of the physical workspace as a virtual barrier tape (see Figure 3). It informs the user about the boundaries of his walking workspace defined by the tracking area or the display devices. It also enables the user to intuitively navigate in the virtual world, without a break in presence, by "pushing" on the virtual barrier tape. Moreover, even if they do not display the user's motion workspace in the virtual environment, previous work about natural walking also has to consider these workspace to prevent the user from colliding with the real environment or leaving the tracking area. Thus, they can determine when the user reaches the limits of the workspace to achieve redirected walking techniques [20] or resetting techniques [25] such as the Freeze-backup or the 2:1 Turn.

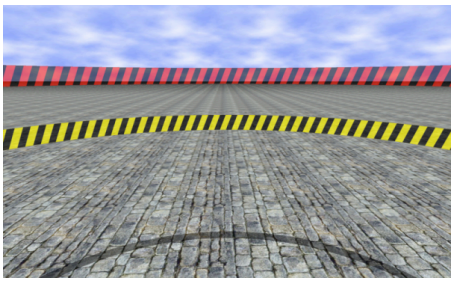

Figure 3: The "Magic Barrier Tape" displays the limits of the user's mobility workspace. (Image from G. Cirio, courtesy IRISA.)

Additionally to the user's motion workspace, other workspaces (sound, visual, interaction and haptic, etc.) have to be considered in the virtual environment. For example, the "bubble" technique [7] proposes to display the limited workspace of a haptic device by a semi-transparent sphere that surrounds the manipulated cursor (see Figure 4). When the cursor is inside the "bubble", its motion is position-controlled. However, when the cursor is outside, the user can move the "bubble" into the virtual world using a velocity control. For prop-based haptic interaction [19], a real object (prop) is linked with a representation and an action in the virtual environment. So the haptic device workspace has to be embedded in the virtual environment to co-locate the prop with this virtual representation and to determine the user's interaction area. Moreover, the Hand Held Display (HHD) [1] is a LCD display whose position and orientation are captured by a tracker. This display can be seen as a window on the virtual world. The system needs to know

the location of this visual workspace according to the user's location to compute the user's frustum. With the prop or the HHD, the co-location between the real world and the virtual world has to be maintained even if the user navigates or changes his scale in the virtual environment.

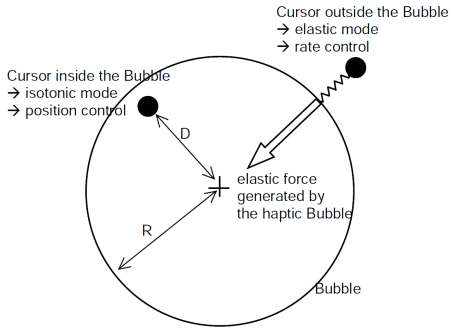

Figure 4: The "bubble" displays the limits of the haptic workspace. (Image from L. Dominjon, courtesy University of Angers.)

Within collaborative virtual environments, users must be able to communicate in order to perform closely coupled collaborative tasks. However, comprehension problems can occur for users with different viewpoints on the virtual world [10]. Even if they can see each other user's avatar, its position, and its orientation in the virtual world as in CALVIN [16], users have difficulty in perceiving what the others see, and more generally what they are doing and what they can do. To overcome these perception problems, Fraser et al. [10] explicitly outline each user's view frustum using a wire-frame model. This model has to be adapted according to the users' display device. Moreover, when a user interacts with an object, they represent the link between this user and the manipulated object by extending the avatar's arm to the object. By extension, the *spatial model of interaction* proposed by [2] can be seen as an interesting approach to describe users' multi-sensory perception. This spatial model defines sensory focus and nimbus for each user. The focus corresponds to the area in which a user has a sensory perception of the other users or of the virtual objects. The nimbus corresponds to the area in which the others have a sensory perception of this user. The focus and nimbus can have different sizes and shapes according to their corresponding media (sound, visual, haptic, etc.). These awareness areas are not necessarily symmetrical. Even if focus and nimbus do not represent users' real environment, they are directly linked to the features of this environment. Moreover, users carry their focus and nimbus when they move in the virtual world.

2.2 Software Models for VR System Design

Lots of VR software models have been proposed to support the development of virtual reality applications. Even if the first ones are very specific to particular technical features (scene-graph, operating system, input and output devices, etc.), some models such as VR-Juggler [3] aim to be independent from the operating system and VR hardware configurations. VR^{2}S [24] can also support multi-sensory output and various input paradigms. This makes it possible to run VR applications in desktop environments by simulating VR devices with standard desktop input paradigms. Additionally to this device abstraction, rendering in VR^{2}S can be performed with several low-level APIs such as OpenGL or ray-tracing systems. As stated by Ohlenburg et al. [18], these device abstraction layers provide a solution to deal with a large variety of interaction devices, but are usually limited to a specific set or type of input devices. So they introduce DEVAL as a generic device abstraction layer that defines device classes and structures them hierarchically. Thus, it can be easily extended by new device types. However, all these abstractions consider devices only as input or output data, and not as real objects with their associated physical workspace to embed into the virtual world.

Robinett et al. [21] propose a hierarchy of coordinate systems to model a Head Mounted Display (HMD) system. This hierarchy of coordinate systems enables the system to modify the user position, orientation and scale in the virtual environment and to maintain the relationship between the real world and the virtual world using transform compositions. Dive [13][11] defines also its own data-structure, which describes a scene in the manner of a scene-graph. Dive and Diverse [14] aims at creating extensible, reconfigurable and device independent virtual environments, through device abstraction and modular design. In this context, a program can run on different hardware configurations using appropriated configuration files. Dive stresses that VR applications should be adaptable to hardware and low-level software changes. In particular Dive proposes high-level concepts such as Vehicle, User and Body Abstraction [23] in order to manage different hardware configurations. Simple Virtual Environment (SVE) [15] also allows to design VR applications independently from the system configuration at run-time. Its design allows a variety of device configurations to be specified at run-time by providing a separation between the devices used and the environment model, and by defining how the device input affects the model. It also makes it possible to configure the software in order to use the best possible devices at run-time. SVE makes a separation between the VE model and the physical devices used, so it is easily configured according to the I/O devices used. The VE model includes a description of the 3D scene and a model of the user. This user model can be driven by input devices, and it can drive output devices. However, none of these VR software consider the physical devices as explicit 3D volumes that virtual representation could be embedded within the virtual world.

For collaboration, Zhang et al. [27] propose a similar approach using the Java 3D scene-graph [22]. The Java 3D `ViewPlatform` concept uses a particular coordinate system to model the users' display device, with the idea of *"Write once, view everywhere"*. Even if this `ViewPlatform` makes it possible to adapt a VR application to several display devices, it cannot represent the users' visual workspace in the virtual environment. Moreover, it is not able to deal with other sensory workspaces.

Mulder et al. [17] describe a first notion of sensory workspaces. For a particular immersive device (a mirror-based display), they model the user's visual space and the user's interaction space. The visual space is defined according to user's head position in relation to the display device position. The interaction space, where a user can perform direct 3D interaction, is limited to the area that the user can reach. They define the user's "direct workspace" as the combination of these two spaces. Several of these personal immersive devices can be placed side by side to enable several users to collaborate in a "*physically shared* workspace". However, each user seems to be unable to freely navigate (i.e. to move his "direct workspace") in the virtual environment because the spatial relationship with the others has to be maintained. Moreover, this solution does not enable users to have remote collaboration and to visualize the others' "direct workspace" in the virtual world.

2.3 Synthesis

Many kinds of VR applications require the users' physical environment to be embedded into the virtual environment. This embedding aims to simply model or to represent this real environment into the virtual world. Modeling users' physical environment improves user presence by matching the virtual world with the real world and by providing an environment safe from collisions or tracking problems. Representing the boundaries of users' physical workspaces enables users to be aware of their interaction capabilities (or the interaction capabilities of the other users in the collaborative case).

However, each VR application achieves a particular embedding of users' physical environment to meet its requirements instead of proposing a generic software model.

Some VR software models propose a device abstraction layer to enable developers to design applications by simplifying the integration of various input or output devices. Moreover, they propose also to specify the devices configuration at runtime in order to adapt the software to hardware devices with no additional programming. However, they do not deal with the representation of these devices in the virtual environment, and they can neither describe the spatial relationships between these physical devices, nor model the users' physical workspace associated to each device.

Other solutions describe the organization of users' physical environment by a hierarchy of coordinate systems and introduce the notion of workspace, but they do not consider the physical workspaces of a user as explicit 3D volumes. Moreover, these approaches depend on the system properties such as the scene-graph, the system architecture, etc. Nevertheless, these can be seen as a first basic hierarchy of workspaces, even if they do not propose software models for embedding multi-sensory workspaces associated to various physical devices.

Last, the notion of workspaces introduced by Mulder et al. [17] must be generalized to all the sensory workspaces and to various devices. It must also maintain the spatial relationships between workspaces even if users navigate or change their scale in the virtual environment in order to combine several interaction techniques.

3 Overview of the IIVC

We need a generic solution that considers the users' physical environment during the VR software design, its deployment and its use. This solution must make the link between these three steps: it must propose a high-level model to describe, configure and modify the users' physical workspace organization whatever the immersive devices used.

3.1 The hierarchy of workspaces

We propose to model the users' physical environment as a structured hierarchy of virtual workspaces. We define the motion workspace as the area where a user can move his body. The visual workspace is not limited to a display device but to what the user can see through and around such a device. A sound workspace represents the area in which a user perceives sound. An interaction workspace is the area where a user can interact. A haptic workspace is the area where a user can interact and have feedback when he uses a haptic device. We call *stage* the reference workspace of our hierarchy (see section 4.1 for more details about this structure). Each virtual workspace must be described and located in relation to this *stage* or to another workspace included in the *stage*. Thus, we obtain a structured hierarchy of workspaces that depicts the real-world spatial relationships between these workspaces. Each workspace can contain real or virtual objects according to its sensory features, such as a tangible interface co-located with a virtual tool [19].

3.2 The IIVC Concept

We propose the Immersive Interactive Virtual Cabin (IIVC) concept as a generic model to describe and manage the relationships between users, their physical environment, the virtual environment and the VR software developers. The IIVC is a link between the real world and the virtual world, but it can also be seen as a link between the end-users and the VR software developers (see Figure 5).

Coexistence End-users are located in the physical environment, so they can act on the real objects and on the input devices.

Design Developers create the virtual environment and choose which interaction and navigation techniques will be used.

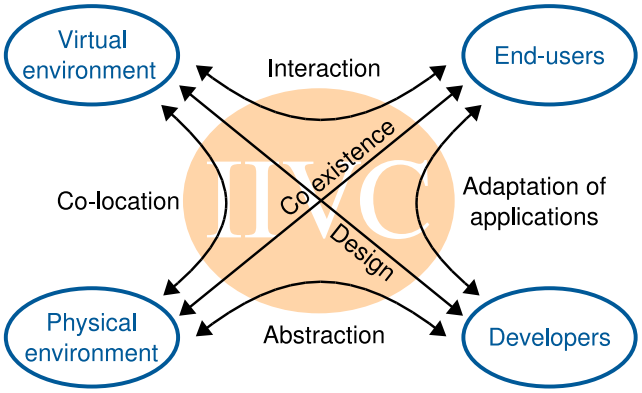

Figure 5: The IIVC concept.

Interaction End-users perform navigation and interaction tasks in the virtual environment. Sometimes, they have to perform these tasks in collaboration with other users. A good presence experience in the virtual environment enables them to interact more effectively.

Co-location 3D spaces of the virtual environment match 3D spaces of the physical environment to link real objects with their representation and action in the virtual world.

Abstraction Developers can design VR software with an abstraction of users' physical environment, which makes VR software more generic.

Adaption of applications Developers can efficiently configure and adapt applications to end-users' real environments.

4 The IIVC Model

This section describes the structure and the main operators of the IIVC model.

4.1 The IIVC Structure

The IIVC can be defined as an abstraction of the users' physical environment in the virtual world. It enables developers to implement their VR software without considering the physical devices used. For example, developers only have to manage position, orientation and scale of each user's IIVC when they develop navigation techniques. In a second step, each IIVC is configured with the features of each user's physical devices (size, shape, hierarchy of workspaces). The IIVC is based on three main components: the workspace, the *stage*, and the *conveyor*.

The *stage* is a virtual description of the users' real environment. It usually matches the room where users interact, but it is also the virtual space containing the virtual representations of users' workspaces. These workspaces are defined by the features of the physical devices used. For example, motion workspace limits are often defined by the boundaries of the area in which users can move: position of the display devices (like in a CAVE^{TM}or a Reality Center) or limits of the tracking area. These workspaces are organized in a hierarchy of included 3D spaces into the *stage*. Each workspace has its own 3D shape and its own coordinate system to locate smaller workspaces or objects (real or virtual) that it contains. The *stage* uses its own coordinate system to locate directly or indirectly all the users' workspaces and all the objects of the IIVC. With this organization, the IIVC model is able to deal with physical reconfiguration such as modifications of workspace position and shape, additions of new screens or other devices, etc.

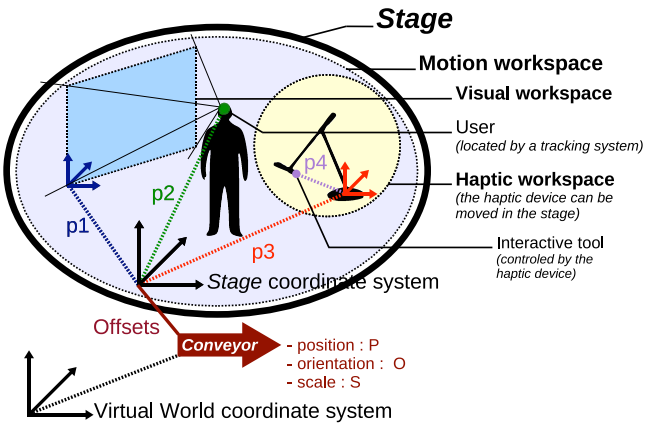

Figure 6: The IIVC structure: the *conveyor* carries the *stage* with its *workspaces* in the virtual world.

The *conveyor* is the integration frame of the *stage* into the virtual world. This *conveyor* is located in the virtual world coordinate system, so it has its own position, orientation and scale in this world. The *stage* is linked to the *conveyor* with position, orientation, and scale offsets (see Figure 6). The *conveyor* also defines the navigation technique, the travel direction, the rotation center, and the scale of the IIVC. So the *stage*, its workspaces and consequently the objects inside the workspaces are carried by the *conveyor* when it moves or changes its scale in the virtual world.

The *conveyor* is totally virtual, while the *stage* makes the link between the real world and the virtual world. With this splitting into two parts, we have to decide where to put the limit between the *stage* and the *conveyor*. In other words, we have to choose which part of the real world must be embedded in the virtual environment. Indeed, we cannot represent all the real world in the virtual environment for all users. So we propose to define the limit of the *stage* as the last physical level which cannot move during the simulation. For example, in a CAVE^{TM}, the limit of the *stage* will be the cube defined by the screen's position. However, if the user interacts on a mobile platform such as a flight simulator, the limits of the *stage* will be the space surrounding the platform.

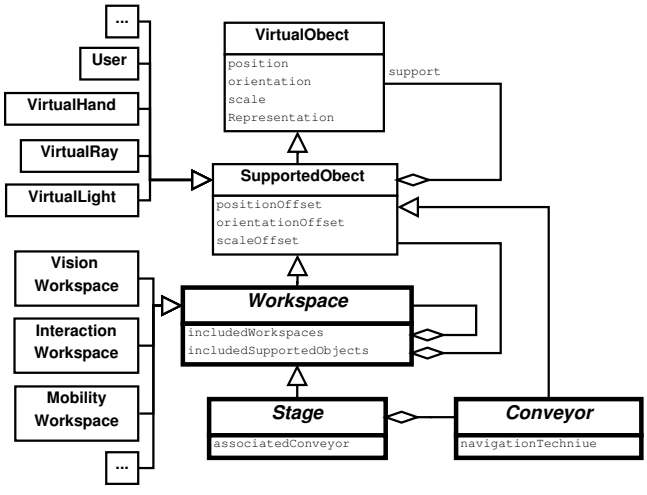

Figure 7: Partial UML model of the IIVC.

Like in Dive [11], we propose to manage our own data-structure in the manner of a scene-graph, without direct dependence to the 3D scene-graph that we use for our 3D graphic visualization. This data-structure rely upon the `SupportedObject` component: it is a `VirtualObject` that can be attached to a support, it can be compared to the famous `Transform` VRML Node. This generic component is provided with mechanisms ensuring the proper propagation of data updates between components at run-time, in order to make it able to compute its state relative to its support state. We use the `Observer` design pattern (GoF293)[12] to propagate the changes from a support towards its supported objects.

The IIVC software architecture is based on a central component: the `Workspace` (see Figure 7). This component is an extension of the `SupportedObject` with new functionalities described section 4.2. The `Workspace` is able to manage virtual objects such as virtual lights, virtual rays, virtual viewing frustums and to include other workspaces. The `Stage` is a particular `Workspace` that is linked to a `Conveyor`. The support of a `Stage` should always be a `Conveyor`, this is why we choose to make a special link between these two classes. The `Stage` is the root of hierarchy of the users' physical workspaces. Lastly, the `Conveyor` describes the navigation technique used.

All the virtual objects that will be embedded in the IIVC inherit also from the `SupportedObject`, such as the `User`, the `VirtualHand`, the `VirtualRay` or the `VirtualLight`. Each of these classes comes with its own behavior and functionalities that we will not detail here.

In the same way, specialized workspaces such as the `VisionWorkspace`, the `InteractionWorkspace` or the `MobilityWorkspace` inherit from the `Workspace` and come with their own behavior and functionalities. For example such workspaces will require at least to know the relative location of their associated `User` in order to adapt the view to his position or to make him aware of some interaction possibilities or constraints (see Figures 8 and 9).

Last, the IIVC software architecture makes it possible to describe the users' physical environment independently from the usual 3D scene-graph or the 3D graphics API used to visualize it. It allows to switch from one 3D graphics API to another one without changing the core of our VR software, but only the component in charge of the coupling with the 3D visualization. This architecture can be described in a configuration file and its components can be associated to external 3D graphical representations. These graphical representations can be described using languages such as X3D or Collada without modifying the IIVC software components.

4.2 The IIVC Operators

The operators are the basic and generic operations that are used to manage the IIVC structure. We provide a library that enables VR developers to implement several functionalities as described in the section 5. First, the basic operators are:

Bo1: modify the position (6 DoF) or scale of a `VirtualObject`,

Bo2: modify the features of a `VirtualObject` (range of a virtual light or a virtual ray, etc.),

Bo3: provide a new support to a `SupportedObject`,

Bo4: modify the offset values of a `SupportedObject`,

Bo5: add or remove a `VirtualObject` into a workspace,

Bo6: provide a new `Conveyor` to a `Stage`,

Bo7: compute the local or global position of a `SupportedObject` in relation to another frame.

Second, we provide higher level operators, obtained through combination of basic operators, such as:

Ao1: superpose several `Stages` or several `Conveyors`,

Ao2: provide the same `Conveyor` as a support to several `Stages`,

Ao3: link a `Conveyor` to a `VirtualObject`,

Ao4: detect the proximity of `VirtualObjects`,

Ao5: compute the intersection of `Workspaces`,

Ao6: modify the shape of a `Workspace` (for example the virtual frustum associated to a `VisualWorkspaces`),

Ao7: restrain DoF for position modification.

This set of seven high-level operators does not pretend to cover all possible operations in a VR, it will have to be extended in the future.

5 The IIVC main Functionalities

The IIVC concept provides several functionalities to VR application designers in order to optimize end-users' navigation, interaction, presence and collaboration according to their physical environment. We analyze the IIVC main functionalities from both an end-user's point of view and a developer's point of view. The IIVC enables VR developers to integrate many VR functionalities proposed in the literature, and also to introduce new VR functionalities thanks to its particular architecture.

5.1 Navigating with the IIVC

5.1.1 Navigation from an End-User's Point of View

Users can move within the motion workspace included in their *stage*. If a user can be located in this workspace (with a tracking system), his view frustum must be distorted according to his head position (head-tracking). This visual workspace can be seen as the *stage* "windows" on the virtual world. It enables users to observe or to position themselves in an intuitive way in the virtual environment.

Users can use almost any navigation technique or metaphor proposed in the literature in order to move their IIVC. For example, they can "fly", "walk", "teleport" themselves, turn around a position or an object, join or follow another user or another object, etc.

5.1.2 Navigation from a Developer's Point of View

Some of these navigation facilities, such as real or virtual walking, flying and teleportation, are provided by directly using some basic operators (**Bo1**, **Bo4**) of section 4.2.

Higher-level functionalities are obtained by combining these operators. For example, allowing a user to select an object in order to turn around it, can be realized by providing this object as the new support of his conveyor (**Bo3**, **Ao3**) (with a null translation offset), computing the new translation offset of the stage according to the current positions of the virtual object, the conveyor and the stage (**Bo7**, **Bo4**), and restraining navigation interaction to only the rotation of the conveyor (**Ao7**). Joining or following a user or an object can be achieved in the same way.

5.2 Carrying 3D Interaction Tools

5.2.1 Interaction from an End-User's Point of View

3D interaction tools such as a virtual ray or a virtual hand can be included in the users' workspace as particular real or virtual objects. So, as in 3DM [5], these interaction tools are automatically carried by the IIVC when it moves or changes its scale in the virtual world. Moreover, users can organize their workspaces by placing these interaction tools according to the kind of interaction they have to perform.

5.2.2 Interaction from a VR Developer's Point of View

It is easy for a VR developer to combine such interaction techniques with navigation, because he can locate these interaction tools in relation to the *stage* coordinate system of the users' workspaces (see Figure 6) (**Bo3**, **Bo4**).

5.3 Making Users Aware of the Physical Environment

5.3.1 Awareness from an End-User's Point of View

A user and the real objects located in his physical environment can be embedded in the virtual world through the *stage*. So the user and these real objects can be co-located in the real and virtual world as in [19], even if the IIVC is moved or scaled in the virtual world.

An IIVC makes the user aware of his interaction capabilities and limitations by representing the limits of his workspaces (by a visual representation, by a sound, etc.). For example, it is possible to light up the virtual objects located in a user's interaction workspace in order to show him which objects are reachable (see Figure 8).

Virtual "semi-transparent" glasses can also prevent users from crossing the boundaries of the tracking system or colliding with the display device (in a full immersive device). When the user inside the immersive device is far from the limits of his motion workspace, the glasses are totally transparent. When the user comes closer to these limits, the glasses become visible to avoid the user crossing the workspace limits and breaking his presence (see Figure 9).

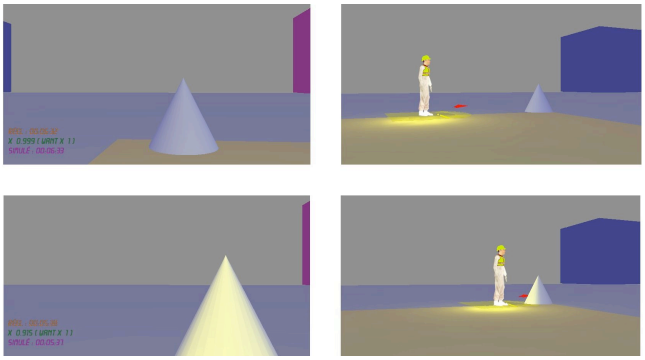

Figure 8: When objects enter the yellow user's interaction workspace, they are illuminated by a colored light (pictures on the left). The other users can also see that objects enter the yellow user's interaction workspace (pictures on the right).

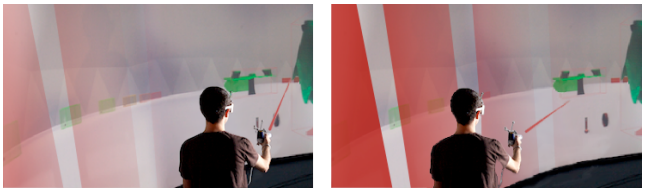

Figure 9: The closer to the display device the user comes, the more visible the "semi-transparent" glasses become, to avoid collision with the physical workspace.

5.3.2 Awareness from a Developer's Point of View

Users' physical workspaces are limited by walls, boundaries of tracking systems, display devices, etc. By embedding these workspaces into the virtual environment as 3D volumes, the IIVC makes the virtual world match the real world, and it makes it possible to perform some computations, for example collision detection between users and workspaces (**Ao4**, **Ao5**), or virtual match-

ing such as adjustment of the range of a virtual light to the interaction workspace geometry and size to highlight all accessible objects (**Bo2**, **Ao5**).

5.4 Collaborating through several IIVC

5.4.1 Collaboration from an End-User's Point of View

IIVC navigation lets users progress independently from the other users in the virtual world: users can have heterogeneous perspectives of the virtual world, which can be very effective for collaborative work [4].

The IIVC provides functionalities to improve the collaboration between users, such as joining or making collaborative navigation together, or interacting with the *conveyor* or the *stage* of another user in order to move, rotate, or scale it.

An IIVC can lap over another one: it establishes a relationship between different real environments through the virtual world. It has some restrictions because the real objects cannot appear in each real environment. However, a real object in an IIVC can have a virtual representation in the virtual world, so it can appear virtually in the other IIVC.

The IIVC represents users inside their physical devices in the virtual world in order to make users aware of the other users' interaction capabilities, which can improve the collaboration. It can help them to understand what the others are doing and where they are looking (see Figure 1), which physical device they are using, which object they can reach without performing a navigation task, etc. For example, Figure 8 shows the yellow user's interaction workspace lighting up to inform other users about which objects this user can reach.

5.4.2 Collaboration from a Developer's Point of View

As in Robinett et al. [21], when a user performs a navigation task, the VR developer can choose to move, rotate, or scale the IIVC rather than the virtual world, which allows each user to have their own viewpoint (**Bo1**, **Bo4**).

Synchronization of several users can be realized by setting their *conveyor* at the same position (**Ao1**) or by linking their *stages* to the same *conveyor* (**Ao2**).

Considering that the IIVC represents its user's physical environment in the virtual environment, it can also be considered as an interactive object, which allows a VR developer to offer interaction with an IIVC of other users (**Bo1**, **Bo4**). It also naturally makes it possible to overlap several IIVC (**Ao1**) to provide collaborative awareness.

6 IIVC Applications

The IIVC model has been implemented as a set of reusable modules, and all users' physical workspaces can be described within a file that we call the configuration file.

First, we present how we have used the IIVC to design several and implement multi-scale collaborative virtual environments. Then, we discuss how existing VR techniques could be designed using the IIVC model.

6.1 First Instances

To demonstrate the possibilities of the IIVC, we have seamlessly integrated several classical interaction techniques in the IIVC such as virtual ray, and several collaboration facilities such as techniques to meet others, to navigate with the others or to leave 3D annotations (for example, a landmark representing an interactive viewpoint). All these tools can be carried by the user in his IIVC.

We thus obtained multi-scale collaborative virtual environments [9] that we have tested with simple workstations, with a Reality Center (an immersive device with stereoscopic vision, head-tracking and an area in which the user can move) and with a workbench. The IIVC model adapts our applications seamlessly to these kinds of devices: we have just to change the application configuration files. It also enables different users to interact in the same virtual environment with different devices: we have tested our collaborative applications with one user in a Reality Center and two other users in front of simple workstations (see Figure 1).

6.2 Instances of "State of the Art" VR Techniques

To illustrate the use of the IIVC model, we discuss how this model could be useful to design three "state of the art" VR techniques. These techniques involve a motion workspace for the first one, a haptic workspace for the second one, and a movable visual workspace for the last one.

6.2.1 Limited Motion Workspace

The "Magic Barrier Tape" [6] (see Figure 3) could be implemented using the IIVC. The virtual barrier tape would be displayed just inside the real limits of the motion workspace, and this motion workspace would be directly included in the *stage*. As long as the user stays in this delimited area, he can freely walk to navigate in relation to the *stage* (**Bo4**). But, when he pushes on the virtual barrier tape, spatial movements of the *conveyor* would be computed from this action on the barrier tape (**Bo1**, **Ao7**). Thus the user could perform long-distance navigation in the whole virtual world by moving the *conveyor*.

6.2.2 Limited Haptic Workspace

The "bubble" technique [7] (see Figure 4) could also be implemented using the IIVC. The limited workspace of the haptic device would be represented by a concentric sphere that would be slightly smaller than this workspace. This haptic workspace would be included in the global motion workspace that would be also included in the *stage*. As long as the 3D cursor associated to the haptic device stays in the "bubble", it would be used for interaction within the *stage* as usual. But, when the cursor goes outside this area, it would be used for navigation: spatial movements would be computed from the cursor position in relation to the "bubble" boundaries (**Ao4**, **Ao5**). These movements can be used to move the *conveyor* (**Bo1**) in the virtual world in order to maintain the co-location of the "bubble" with the haptic device.

6.2.3 Movable Visual Workspace

A Hand Held Display (HHD) [1] could also be modeled using the IIVC. The visual workspace associated to the screen would be defined as a movable workspace included in the user's motion workspace. As the user's head and this visual workspace would be located in the motion workspace, it would be easy to compute the visual workspace deformation (a viewing frustum) (**Ao6**) and the image to display on the screen. The way to compute this deformation would stay the same even if the user navigates by moving his *conveyor* and consequently the whole IIVC in the virtual world.

7 Conclusion

With immersive devices, the interface between a user and a virtual reality software is not restricted to a flat panel, but must consider a full 3D space. Several multi-sensory 3D workspaces are located in this 3D space: each workspace is dedicated to a functionality like visualization, tracking, interaction, etc.

With its ability to manage a hierarchy of 3D workspaces, the Immersive Interactive Virtual Cabin (IIVC) provides a generic software model to embed users' physical workspaces in a virtual environment. The IIVC is an abstraction of immersive devices which enables the VR developers to design applications without taking which immersive devices will be used into consideration. It can be adapted to a simple workstation or to a full immersive device like a CAVE^{TM}. It matches the real world with the virtual world to maintain head-tracking of users or co-location of real objects even

if users navigate in the virtual world. This navigation (position, orientation and scale changes) is independently performed by each user and is also applied to the interaction tools included in the users' workspaces.

The IIVC software architecture makes it possible to describe the users' physical environment independently from the 3D graphics API used to visualize it. Only one component is in charge of the coupling with the 3D visualization. This architecture is described in a configuration file and its components are associated to external 3D graphical representations. So, like with Dive, Diverse or SVE it is easy to adapt at run-time the VR software to a specific hardware configuration. The advantage of the IIVC is that it also visualizes the physical features of the hardware input and output devices.

Last, the IIVC is useful to Collaborative Virtual Environments (CVE) developers because it automatically provides a 3D representation of a user's physical workspaces to the other users who share the same CVE, making them naturally aware of the physical activity and limitations of each other user.

8 Future Work

As we have essentially used visual and motion workspaces, now we need to explore other kinds of workspaces such as sound or haptic workspaces.

We also need to enhance our existing model with other high level operators, especially for the perception of the users' interaction capabilities and for collaboration. We will have to evaluate how much embedding users' physical workspaces within the IIVC can enable the user to better understand their interaction capabilities or limitations. In collaborative situations, it should naturally provide a better awareness of the other users' activities and interaction capabilities.

Finally, we need to propose a standardized formalism to describe workspace management and manipulation. Thus, a language to describe the physical workspace of each user and its mapping with the virtual environment should be defined. This language could be an extension to a language such as X3D or Collada.

Acknowledgements

This work was partly funded by the French Research National Agency project named Collaviz (ANR-08-COSI-003-01).

References

[1] D. Amselem. "A Window on Shared Virtual Environments". *Presence: Teleop. & Virtual Env.*, 4(2):130–145, 1995.

[2] S. Benford, J. Bowers, L. E. Fahlén, and C. Greenhalgh. "Managing Mutual Awareness in Collaborative Virtual Environments". In *Proc. of the Symp. on Virtual Reality Software and Technology*, pages 223–236, 1994.

[3] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. "VR Juggler: A Virtual Platform for Virtual Reality Application Development". In *Proc. of the IEEE Virtual Reality Conference*, pages 89–96, 2001.

[4] D. A. Bowman, S. Coquillart, B. Froehlich, M. Hirose, Y. Kitamura, K. Kiyokawa, and W. Stuerzlinger. "3D User Interfaces: New Directions and Perspectives". *IEEE Computer Graphics and Applications*, 28(6):20–36, 2008.

[5] J. Butterworth, A. Davidson, S. Hench, and M. T. Olano. "3DM: A Three Dimensional Modeler using a Head-Mounted Display". In *Proc. of the Symp. on Interactive 3D graphics*, pages 135–138, 1992.

[6] G. Cirio, M. Marchal, T. Regia-Corte, and A. Lécuyer. "The Magic Barrier Tape: A Novel Metaphor for Infinite Navigation in Virtual Worlds with a Restricted Walking Workspace". In *Proc. of the 16th Symp. on Virtual Reality Software and Technology*, pages 155–162, 2009.

[7] L. Dominjon, A. Lécuyer, J.-M. Burkhardt, G. Andrade-Barroso, and S. Richir. "The "Bubble" Technique: Interacting with Large Virtual Environments Using Haptic Devices with Limited Workspace". In *Proc. of the World Haptics Conference*, pages 639–640, 2005.

[8] T. Duval and C. Fleury. An asymmetric 2d pointer/3d ray for 3d interaction within collaborative virtual environments. In *Web3D'09: Proceedings of the 14th International Conference on 3D Web Technology*, pages 33–41, New York, NY, USA, 2009. ACM.

[9] T. Duval, C. Fleury, B. Nouailhas, and L. Aguerreche. "Collaborative Exploration of 3D Scientific Data". In *VRST'08: Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 303–304, New York, NY, USA, 2008. ACM.

[10] M. Fraser, S. Benford, J. Hindmarsh, and C. Heathq. "Supporting Awareness and Interaction through Collaborative Virtual Interfaces". In *Proc. of the 12th Symp. on User Interface Software and Technology*, pages 27–36, 1999.

[11] E. Frécon and M. Stenius. "DIVE : A Scaleable Network Architecture for Distributed Virtual Environments". *Distributed Systems Engineering*, 5:91–100, 1998.

[12] Gamma E., Helm R., Johnson R., and Vlissides J. *"Design Patterns: Elements of reusable Object-Oriented Software"*. Addison-Wesley, 1995.

[13] O. Hagsand. "Interactive Multiuser VEs in the DIVE System". *IEEE MultiMedia*, 3(1):30–39, 1996.

[14] J. Kelso, L. E. Arsenault, R. D. Kriz, and S. G. Satterfield. "DIVERSE: A Framework for Building Extensible and Reconfigurable Device Independent Virtual Environments". *Virtual Reality Conference, IEEE*, 0:183, 2002.

[15] G. D. Kessler, D. A. Bowman, and L. F. Hodges. "The Simple Virtual Environment Library: An Extensible Framework for Building VE Applications". *Presence: Teleoper. Virtual Environ.*, 9(2):187–208, 2000.

[16] J. Leigh, A. Johnson, C. Vasilakis, and T. DeFanti. "Multi-perspective collaborative design in persistent networked virtual environments". In *Proc. of the Virtual Reality Annual International Symp.*, pages 253–260, 1996.

[17] J. D. Mulder and B. R. Boschker. "A Modular System for Collaborative Desktop VR/AR with a Shared Workspace". *Proc. of the IEEE Virtual Reality Conference*, 0:75, 2004.

[18] J. Ohlenburg, W. Broll, and I. Lindt. "DEVAL - A Device Abstraction Layer for VR/AR". In *Universal Acess in Human Computer Interaction*, volume 4554 of *Lecture Notes in Computer Science*, pages 497–506. Springer, 2007.

[19] M. Ortega and S. Coquillart. "Prop-Based Haptic Interaction with Co-Location and Immersion: An Automotive Application". In *Proc. of the Int. Workshop on Haptic Audio Visual Environments and their Applications*, pages 23–28, Oct. 2005.

[20] S. Razzaque. *"Redirected Walking"*. PhD thesis, University of North Carolina at Chapel Hill, 2005.

[21] W. Robinett and R. Holloway. "Implementation of Flying, Scaling and Grabbing in Virtual Worlds". In *Proc. of the Symp. on Interactive 3D Graphics*, pages 189–192, 1992.

[22] H. A. Sowizral and M. F. Deering. "The Java 3D API and Virtual Reality". *IEEE Computer Graphics and Applications*, 19(3):12–15, 1999.

[23] A. Steed. "Some Useful Abstractions for Re-Usable Virtual Environment Platforms". In *Software Engineering and Architectures for Realtime Interactive Systems - SEARIS*, 2008.

[24] F. Steinicke, T. Ropinski, and K. Hinrichs. "A Generic Virtual Reality Software System's Architecture and Application". In *Proc. of the International Conference on Augmented Tele-existence*, pages 220–227, 2005.

[25] B. Williams, G. Narasimham, B. Rump, T. P. McNamara, T. H. Carr, J. Rieser, and B. Bodenheimer. "Exploring Large Virtual Environments with an HMD When Physical Space is Limited". In *Proc. of the 4th Symp. on Applied perception in graphics and visualization*, pages 41–48, 2007.

[26] C. A. Wingrave and J. J. LaViola. "Reflecting on the Design and Implementation Issues of Virtual Environments". *Presence: Teleoper. Virtual Environ.*, 19(2):179–195, 2010.

[27] X. Zhang and G. W. Furnas. "mCVEs: Using Cross-Scale Collaboration to Support User Interaction with Multiscale Structures". *Presence: Teleop. & Virtual Env.*, 14(1):31–46, 2005.