The Interaction Specification Workspace: Specifying and Designing the Interaction Issues of Virtual Reality Training Environments From Within

Costas N. Diplas[1,2], Achilleas D. Kameas[1], Panayotis E. Pintelas[1,2]

[1] Educational Software Development Laboratory, Department of Mathematics, University of Patras, Greece.
[2] Division of Computational Mathematics and Informatics, Department of Mathematics, University of Patras, Greece.

Abstract Advances in Virtual Reality (VR) systems, Intelligent Tutoring Systems and Agent Technology make it possible to design and develop Virtual Training Environments, where the trainees can immerse themselves and interact directly with the learning domain. This paper presents the Interaction Specification Workspace (ISW) architecture for the specification and design of virtual environments for training purposes. ISW architecture provides the interaction designer with the capability to specify the training interaction with the virtual environment using the Virtual Reality Multi Flow Graph (VR-MFG) as the underlying interaction specification model. ISW architecture implements a design space, where the processes of interaction specification and design of virtual training environments take place inside a three-dimensional virtual environment, the objects of which are tools by themselves. Using the components of such a design space, the designer composes interaction sessions that will be executed when the VR application executes. Current VR development platforms use diverse metaphors for the design and implementation of virtual environments. In order to bridge the consequent semantic gap between designers (the authors of virtual environments for training) and users (the trainees) of such environments, ISW proposes the "virtual programming" metaphor, as the natural evolution of contemporary visual programming: inside a common virtual workspace, the designer can associate the abstract objects (the components of the VR-MFG) with "actual" objects of the target virtual environment (kept in a "world" database) and apply a number of agent templates with training capabilities.

1 Introduction

In Graphical User Interfaces (GUIs) of conventional programs the user "feeds" the application program with data or acts on some data via standard input devices such as keyboard and mouse, and perceives the information through the screen display. In order to do this, a number of interface components (e.g. buttons, list-boxes, etc.) have been invented, that behave as "handles" allowing the end-user to manipulate the data and information processed by the application program. These user interface

components receive the events caused by the user, and display the information produced by the program, serving both the user interface designer, who chooses among a limited number of these standard components to construct the user interface of the application program, and the end-user who learns how to operate on these specific user interface components. Things would be complicated for the end-users if user interface designers used custom-made "controls", which did not fall within the known categories. In addition, every contemporary application program, (e.g. word processing packages, spreadsheets, multimedia tools and authoring systems for educational programs) uses the desktop metaphor as the underlying interaction model through a WIMP (windows-icons-menus-and-pointing) type user interface. In these programs, the *interface* point of view has the leading-role, with the creation of ease-to-use and functional controls and widgets. The *interactivity* point of view follows as a consequence, due to the limited capabilities provided by the standard user interface components which are restrictive for the user-computer dialogue representation. Virtual Reality (VR) applications tend to unify the application program and its user interface: the user interface becomes transparent and can not be distinguished from the "pure" application. VR uses techniques for immersing the user into a computer simulated environment where natural behavior is the interaction paradigm. In a virtual environment (VE) the application functionality and the application interface are not visually or physically separated, but only conceptually. The user interfaces of conventional application programs (even those with Graphical User Interfaces) serve as a representation of that functionality, and are constructed in a way which conveys to the user what the program can do. In the case of VR applications, the term functionality needs to be redefined as what a person can do with the computer program rather than what a computer program has the capacity to do [10]. According to this, the user interface of a VR application must *tell the user what he can do* inside the virtual environment, as well as, *indicate the correct way* in which he must interact with the application program itself in order to accomplish the desired tasks or to achieve the desired goals. To construct user interfaces which satisfy these requirements, the specification and design practices must concentrate more on the *interactivity* issues, rather than on the *interface* issues, since the latter are embedded into the VR applications. Of course, in VR (as well as any other) applications the user does not interact with the entire application, but with specific components of the VE. Thus, in a VR application which is composed mostly of computer-generated three dimensional models, the user is able to navigate, push, move, manipulate or even construct new objects inside the VE. Not all VE objects are interactive but every interactive object needs to be equipped with a number of "controls" which serve as the interface components for a VR application. However, like the physical objects that exist into our surrounding real environment, their virtual counterparts have their own "knobs". For example, all the members that belong to a specific class of objects (e.g. all the doors) can be handled using these common fundamental control "knobs", inside the corresponding virtual environment. This is analogous to the conventional WIMP interfaces: if a window class with one *system-menu*, one *minimize-restore* and one *close button* is specified, then all the windows (members of the same class) are expected to be equipped with at least these controls. In any case, the existence of

these "controls" or "knobs" is not restrictive for the representation of the user-computer dialogue, as it happens in conventional application programs, since these "knobs" refer to the handlers of the objects, and not to the UI widgets. So, the user interface embodied into a VR application ought to be depended on the application domain, rather than the platform used to implement the application. However, during design and development of VR applications the problem arises when conceptual and "abstract" information objects must be visualized.

The problem of bridging the gap between a user's goals and intentions, and the low-level commands and mechanisms required by any application program in order to realize those goals and intentions still exists, despite of the many attempts that have been made, through the creation of effective Uis. The VE designer may find the solution to this problem if he takes advantage of the VR applications' special nature. In general, application programs provide a user interface (even a GUI) as a medium to help both user and system bridge the above mentioned gap. This gap can be expressed in two ways: as the common medium where user and system border on (optimistic view), or as the separating line between the user and the system (pessimistic view). VR applications introduce themselves as a *user interspace* (rather than a *user interface*), a concept which is closer to the optimistic view. This *user interspace* realizes the concept of the interface not simply as a means where a user and a computer system represent themselves to each other, but as a shared context for action where both are agents [21].

Every user's goal and intention in an application program (e.g. modify a database, delete a document, access a site on the web) is being accompanied by the problem: *how this goal can be achieved?* In conventional application programs the solution is provided by the user interface, which must guide the user to accomplish the desired tasks. It is obvious that the user interface has to provide the appropriate widgets and components that must be used by the user in order to accomplish the desired tasks. Concerning VEs, the solution to the same problem is embodied inside the virtual environment itself. The nature of the virtual environment objects shows the user the correct way of interaction, in the sense that the user must perform almost physical actions in order to interact with the VE objects. But, the friendlier the user interface is, the more difficult it is to design it. So, the designers of VEs have to establish unambiguous and robust interaction techniques, providing the user with obvious ways of interaction so that he does not worry about specific controls that exist in the virtual environment but concentrates solely on his specific goals. Moreover, the purpose of VR applications is to provide the user with the capability to perceive with more natural ways the information produced by an application program and interact with complex data. This means that the user's goals and intentions are translated not to abstract actions (e.g. click a button, roll up/down a slider, etc.) but to physical movements, often directly onto the objects of the VE. This forces the VE developers to provide the user with clearly defined goals. As shown and statistically measured in [24], most users focus on what there is to do, and what is already done, inside the virtual environment and not only on how to do it.

So, the VE designer deals with the *interaction design* rather than the *interface design*. Consequently, the specification and design tools for VR applications must provide ways to express interaction, rather than to "build" the interface; this is the major goal to the achievement of which our work attempts to contribute.

1.1 Related Work

The prime effort for the establishment of VR as a new technology and the development of virtual environments was on the high quality graphics that should be supported, the construction of three-dimensional objects and virtual scenes and the development of new hardware devices that could be used for interacting with the constructed VEs [14]. Architectures and systems for virtual world building, VE modeling for various application domains and object manipulation led to systems which combine easy object modeling, creation and manipulation along with (sometimes intelligent) object behavior [12, 30, 2]. Also, approaches which focus on the visualization of physical systems in virtual environments and on the graphical representation of the information retrieval process make use of VE technology as presented in [23], [11].

Recently, interaction design was recognized as an important issue for the implementation of highly interactive VR systems, that exceed a simple 3-D interface, and many attempts have been made in this direction. MR (Minimal Reality) Toolkit [27] is a set of software tools for the production of virtual reality systems and other forms of three dimensional user interfaces. It consists of subroutine libraries, device drivers, support programs and a geometry and behavior description language. Immersive Metaphors project [25], focuses on the design and development of a consistent collection of the immersive techniques and metaphors which would be as powerful and ubiquitous as techniques which are used to build present day 2D Graphical User Interfaces. It is claimed that interface designers could use these techniques and guidelines to build 3D immersive user interfaces quickly and with high quality of interaction. Virtual Reality Interface Toolkit project [6] aims to develop a 3D user interface software toolkit which would provide all necessary 3D widgets, interaction techniques and tools for programmers to design and implement 3D user interfaces for various application domains. A developer can use the provided widgets and techniques or construct custom interface elements by inheriting attributes from existing objects. For the user, the applications built with the toolkit will have a consistent generic 3D user interface, based on a single paradigm that has been evaluated by theoretical and experimental studies. The focus of the Interface Toolkit is to design and evaluate software architectures which allow seamless integration of the 3D interaction techniques and widgets with VR world building functionality. VB2 [15] architecture for the construction of three-dimensional interactive applications proposed that the system's state and behavior are uniformly represented as a network of interrelated objects. The interaction techniques used in this architecture, among others (direct manipulation, gestural input, etc.) included three-dimensional virtual tools, which offered an interaction metaphor to control the VE models' information.

Moreover, the evolution of VB2, the AVIARY [29] architecture proposes that everything that lies inside a Virtual Environment is treated as an *object.* Then each single object which is presented to the user is an *artifact,* and objects that cause the artifacts are called *demons* [28]. Conceptual Design Space [8], attempts to provide virtual tools and 3D interface elements for the construction of VEs. 3D menu systems, widgets, dialog boxes, tool palettes, etc., are used in order to provide a new interaction design metaphor, where users can be able to create virtual worlds, while immersed in one themselves.

The architectures and systems mentioned above are general purpose systems and have not being specifically designed for the development of VEs for training. This application area can exhibit theoretical frameworks and working prototypes [35, 9, 22, 17, 18]. Moreover, almost all of the above mentioned architectures, systems and tools, and most commercial VR development platforms [33, 36] use diverse metaphors for the design and implementation of virtual environments. The designer uses tools with conventional user interface and produces applications with three-dimensional and immersive interfaces. So, the problem which arises is twofold: how to provide the authors-designers with appropriate tools to design the interaction and instructional aspects of the desired Virtual Training Environment, and how to bridge the consequent semantic gap between designers and users of Virtual Training Environments.

The proposed solution is composed of the establishment of a *Petri-Net based graphical formal model for interaction specification with capabilities to represent the instructional aspects of the final application,* and an *architecture which implements a virtual reality tool for interaction specification, proposing the "virtual programming" metaphor, as the natural evolution of contemporary visual programming.* The ISW architecture, along with the underlying graphical formal model is presented in the next section. An example showing the model capabilities is presented in section 3 and the paper concludes with discussion and future work in section 4.

2 The Interaction Specification Workspace (ISW)

The basic components of the Interaction Specification Workspace are the Virtual Reality Multi Flow Graphs (VR-MFG) graphical model and the ISW Architecture.

2.1 The ISW underlying Interaction Specification Model (VR-MFG)

VR-MFG is a graphical formal model that can be used for the specification and design of Virtual Reality Agent-based Training Applications. The model is defined as an extension of IMFG [19, 20], a model used for the specification and design of conventional interactive applications, and incorporates the cognitive features and the powerful analysis techniques of Petri Nets.

Virtual environments due to the three-dimensional graphics they provide, are the ideal platform for representing the elements used by VR-MFG (cubes, spheres, cones, etc.) and the visualization of the concepts these graphical elements reflect. On the other hand, the graphical notation of VR-MFG (mostly adopted from IMFG) makes the specific model one of the most suitable, among other similar approaches [16] for interaction specification in VEs. The usage of a VR environment as interaction specification workspace provides:

- *Reduction of the space* needed for the entire specification due to the capability of the three-dimensional graphical objects to "include" other objects.
- *Supervision* over the specification [34].
- *Definition of different views* and data abstraction due to the "depth" provided by the third dimension.
- *Uniformity* between the design space and the executable one.
- *Establishment of a common context* (the three-dimensional visual representation) for both the author-designer and the user-trainee, enabling the designer to think in user terms.

As a virtual environment consists of *interactive* (active) and *non-interactive* (passive) virtual elements (e.g. objects, concepts, abstractions, information), the VR-MFG which models this VE consists of active (actors) and passive (links) components. Although there is a close relation between the VE's elements and the VR-MFG components, there is an indirect correspondence between them. An active VR-MFG component does not refer directly to an *interactive element* of the VE, but to the task, goal or high and lower level action into which this element is involved. The correspondence between a passive VR-MFG component and a *non-interactive* VE's element, is analogous, since a passive VR-MFG component refers to the visualization of the different information flows that occur in a Virtual Reality interactive application.

Analytically, the components of Virtual Reality-MFG model are:

- *actors* (that correspond to the actions, tasks or goals of the VE's elements), which model the interactive responses that must be performed by the agents that participate in a VE, as a consequence of the occurrence of an event. Events may be caused by other agents, since the user, the objects of the VE and the VE itself, are all agents that act inside a common space. Actors are always preceded and followed by
- *links* (that correspond to the non-interactive VE's elements), which describe the situation that precedes and that results from a user action, through the storage of
- *tokens* (that correspond to the data or/and control information that exist into the VE), which represent abstract data or control structures that are produced or consumed by the VR-MFG components.

All the actors that are ready-to-fire (that is, which may fire after the next event) are maintained in the *actor-ready list* . Moreover, an actor can be viewed as an integral goal, which can be achieved by the satisfaction of a number of sub-goals.

Actors are described by: a *name*, a *set of input* and a *set of output links*, which precede and follow the actor, a *set of firing rules*, which represent the actor's behavior, since the left hand side forms the pre-conditions (that are kept in its input links) and the right hand side includes the post-conditions (that are kept in its output links), a *method*, which represents the lower level actor's functionality, that is, how the actor handles its input data and produces its output ones, and a *type*.

There are four types of actors, namely:

- *Action actors*: they represent a single response which is performed by a VE agent. These actors do not have any VR-MFG represented internal structure. The rules part of each action actor defines how this single task is implemented. VR-MFG does not give a clear description of the task to be done, but specifies the goal decomposition (or the task analysis) in order for the goal to be achieved (or the task to be completed). Action actors define the way the VE's agents interface with the domain-dependent functional core of the application, via single interactive responses.
- *Context actors*: their internal structure represents the task or goal decomposition into sub-tasks or sub-goals, via a number of other context or action actors.
- *Guide actors*: similar to Library IMFG actors, are used to represent the way the task or goal decomposition is achieved. AND and OR decomposition are provided, since these two fundamental actions can model any task or goal decomposition. These two decomposition styles represent a minimal functionality; also, serve the formality and facilitate the conversion of the VR-MFG graphs to a corresponding petri-net. For complex task representation (e.g. task interleaving, task interruption etc.), VR-MFG model provides the *actor-ready list*, the *condition links* and the link *usage properties* (*read-only*, *debit*, *OK*). Of course there is the capability to add more guide actors in the future.
- *Virtual actors*: used for a graphical grouping of actors, without any other significance, but may serve as reusable components during design process.

Links are described by: a *name*, a *set of input* and a *set of output actors* that produce and consume the tokens stored in this link, a *method*, which is performed upon the link's tokens, and a *type*. The definition of link types distinguishes among the type of tokens they store. Each link type stores a specific kind of tokens. This allows the designer to model the different types of information (e.g. data, control) that exist inside a VR Application, and permits system design from alternative perspectives. VR-MFG links are typed so that the different information flows that occur in a VR application are distinguished and each information flow can have its own visualization. The seven types of links, are:

Event links: describe the events that are caused by the agents that participate in the VR application. Users are also treated as agents in VR-MFG, so event links can be used in order to describe any *external* or *internal* communication of events. In VEs, events may be composite, having their own existence, unlike applications that use a

2D GUI, where events can be caused by simple actions (e.g. click, scroll, key-press, etc.). For example, event generation process in numerous cases [7] (e.g. where gesture interaction methods are used), includes an internal structure. Moreover, this is recommended by a number of diverse interaction methods which have already proposed in [5, 31].

VR-MFG provides *event links decomposition*, so the designer can explicitly specify how events can be caused, representing the internal structure of event links.

Perception links: a special kind of event links, which are used in order to represent system responses that are directed to the input-output devices, in order to provide the user of the VR application with the capability to perceive with natural ways special VE responses (e.g. haptic feedback, position orientation feedback activities etc.). These are event links with internal structure, that represent any cognitive perceptual state of the user agent and the tokens they contain are produced and consumed exclusively by the user (or the user agent). *Perception links* are designed in order to make VR-MFG model adaptive to any kind of interaction technique (e.g. direct or indirect manipulation of VR objects, immersion techniques where input is entered via sensors and output is processed by advanced hardware interface devices) and permit platform-independent interaction specification.

Condition links: represent the global or local conditions that precede and result from any agent action that takes place into the VE. Consequently they represent priority of execution and availability of actors. Moreover condition links describe whether any of the VE agents is ready to process another agent's action, that will lead to the achievement of a subgoal, or to the completion of a specific task, into the VE.

Data links: represent the data or control flow, into the VR application. Moreover, they represent the content of messages that pass between the participating agents.

Context links: represent the context, into which a number of interactions are performed. Consequently, context links describe the context to which a specific goal or task belongs and indicate whether a major goal is decomposed into sub-goals, so that a new "session" starts for the accomplishment of each sub-goal. Moreover, context links contribute to the representation of the system's *memory* and *knowledge* issues which are of major importance for any VR agent-based Application. Every actor that belongs to a specific context *knows* the goal of all the other actors that belong in the same context and the VR-MFG can model long-term memory by maintaining the actor ready list and the content of context-out links, since the short-term memory is represented directly inside the current context.

Communication links: model the effects that the separate micro-worlds existing inside the entire VR application may have on one another, since there exists a separate VR-MFG for each virtual micro-world.

Learning Links: represent the learning issues which rule the training interactions and the student-trainee dialogue. These links are founded upon alternative instructional

strategies and provide representation of autonomous agents with instructional capabilities [26].

VR-MFG, being graphical tool, has its special symbolism which is shown in Figure 1.

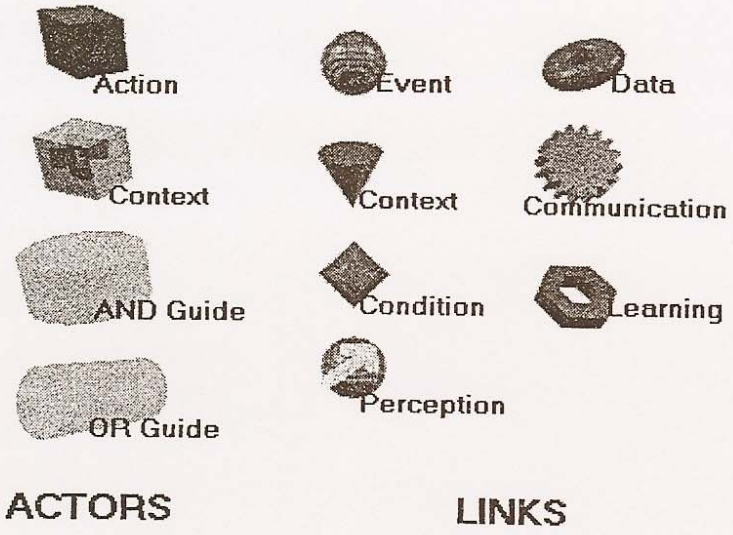

Fig. 1 VR-MFG symbolism

2.2 ISW architecture

The Interaction Specification Workspace is a software architecture for the specification and design of 3D virtual training environments. The overall system structure includes three major modules: The Prototype Interaction Editor (Practor), The Analysis and Evaluation module and the Prototyping facility; the application content, is an external entity which belongs partially to ISW since the pure content (objects, lights, real-time graphics etc.) can be produced by a third party VR development tool. The system structure is presented in Figure 2.

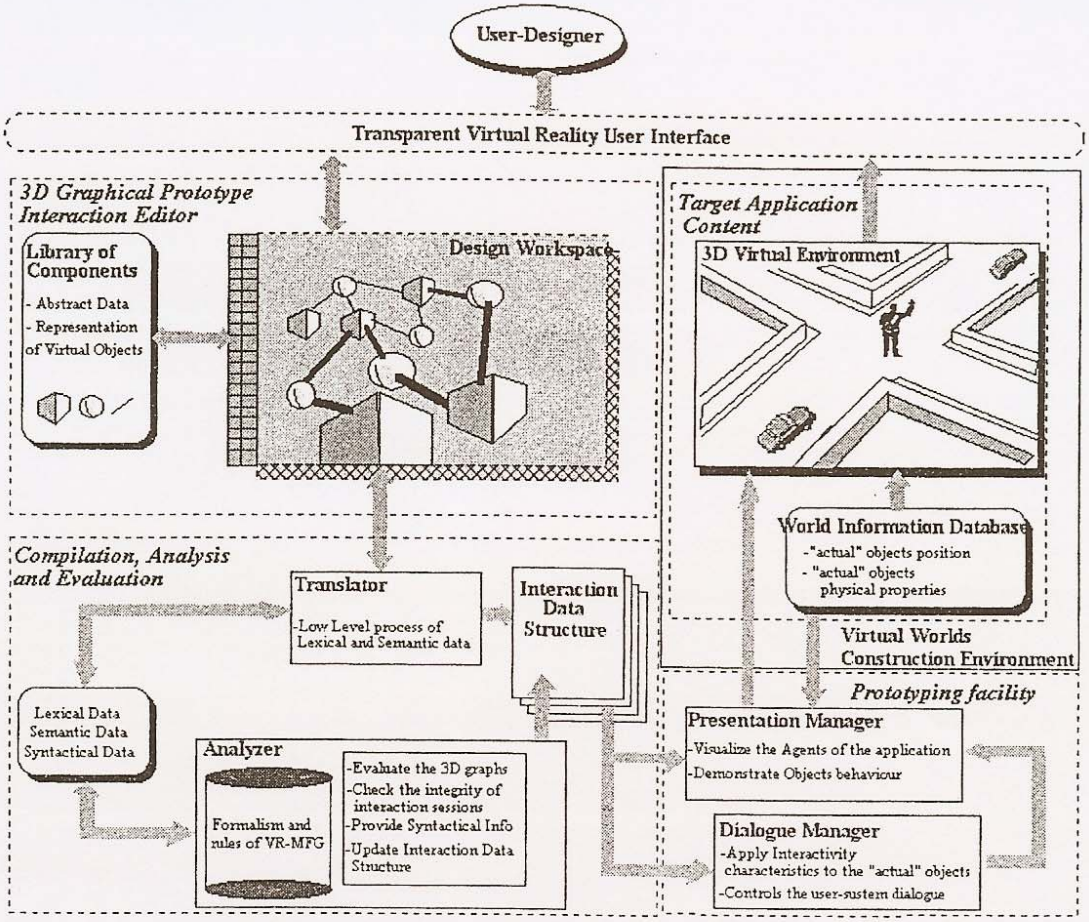

Fig. 2 Overall structure of ISW

ISW provides the interaction designer with the capability to use the Prototype Interaction Editor in order to construct the graphs that represent the interactive and training sessions of the application. The workspace of the Prototype Interaction Editor is an abstraction of the virtual tutoring environment which is constructed. The actual objects contained into the final virtual environment have their own abstract representation into the workspace, enabling the designer to think and design in user terms.

VR-MFG, described in the previous section, is used as the underlying ISW model. The translator processes the interaction graphs at a low-level of lexical and syntactical detail. The analyzer evaluates the correctness of the produced graphs and the integrity of the interaction sessions, and produces and updates the interaction data structure. The analyzer can also be used as a wizard, which can guide the designer during the authoring process.

The prototyping facility enables the user to swap between the 3D graphical Prototype Interaction Editor and the virtual environment which is under construction. This enables the designer to take the place of the user-trainee and check if the application

meets its initial interaction and learning specification. At this point, the designer is able to evaluate the results which are related with the interaction and instructional design rather than the results related with the functionality of the entire application.

The tools of ISW can be used as "add-ins" utilities for a third-party VR development environment, extending the capabilities of this environment. The author-designer uses this VR development tool in order to construct the "pure" application content. According to the methodology the ISW proposes, he is able to use the *3D Graphical Prototype Interaction Editor*, in order to specify the interaction aspects of the final virtual environment. The *Analysis and Evaluation module* checks the graphs for correctness and the specification is recorded in the *Interaction Data Structure*. This data structure along with the *Dialogue and Presentation Managers* "tie up" the ISW and the VR development environment, in order to form a loosely coupled system. The interaction specifications of Interaction Data Structure are applied on the objects of the *World Information Database* which is updated via the VR development environment.

3D graphical Prototype Interaction editor. The designer uses a library of 3D screen components that represent the components of the VR-MFG and composes interaction graphs. The 3D graphical Prototype Interaction Editor, is independent from the target application content, and the designer can associate the abstract VR-MFG components with "actual" objects of the target virtual environment. The 3D design workspace is a virtual environment by itself, its objects are of some standard kind of shape (cube, sphere, cylinder, line, etc.), and do not need to have gravity and velocity attributes. However, the user-designer can:

- move and rotate the objects, setting their position, orientation and placement,
- add colors to them, emphasizing to special sub-graphs,
- group them into bigger structures, reducing the overall graphs complexity
- isolate groups of them as individual graphs and place them back to the library (reusability of library components),
- associate them (individually or as groups) with the "actual" objects that participate into the target application.

Analysis and Evaluation module. The Analysis and Evaluation component is also application independent. Its basic components are:

- the translator, which is responsible for processing the composed 3D graphs at a low level of lexical and semantic detail. The lexical and semantic data that result are further processed by
- the analyzer which is responsible for the evaluation of the correctness of the produced graphs, the integrity of the constructed interaction sessions, and the modification of the interaction data structure. This data structure contains the interactivity and behavioral characteristics of the objects (e.g. data about the types of the objects, the tasks that are performed by these objects, the context, the user goals and intentions, the driver intelligent agents that are associated with specific objects, etc.).

Prototyping Facility. The Prototyping Facility is composed of:

- the Dialogue Manager, which uses the interaction data structure in order to form the dialogue between the user and the objects of the target application or between the objects themselves, is responsible for the application of the intelligent and interactivity characteristics of the "actual" objects of the target application, and the dialogue control during the prototyping process (messages, information boxes, etc.), and
- the Presentation Manager, which interrelates the intelligent and interactivity characteristics of the objects with the physical properties (position, orientation, size, etc.) of the "actual" target application's objects. The former are provided by the interaction data structure and the Dialogue Manager and the latter by the World Information Database which is constructed and updated via the design editors of the third-party Virtual Reality Development Tool. The Presentation Manager is also responsible for the visualization of the agents that participate either as stand alone entities or as driver programs which are associated with the user or with other objects of the target application, and the demonstration of the objects behavior as they interact with each other or with the user through the "transparent" VR user interface.

3 An example application

Any VR Application can be viewed as a collection of different VEs (micro-worlds) that constitute the entire application. These multiple worlds may be concurrently active, just like a conventional GUI application, where several windows are open and running different applications, but only one can have the user focus. The aim of ISW, is not to specify the concurrency for the entire application, but to split the design effort for concurrency, among these multiple worlds, via the use of VR-MFG.

The designer is provided with the capability to associate the abstract objects (the components of the VR-MFG) with "actual" objects of the target virtual environment kept in the object database, and apply a number of agent templates, inside a common virtual workspace (Figure 3).

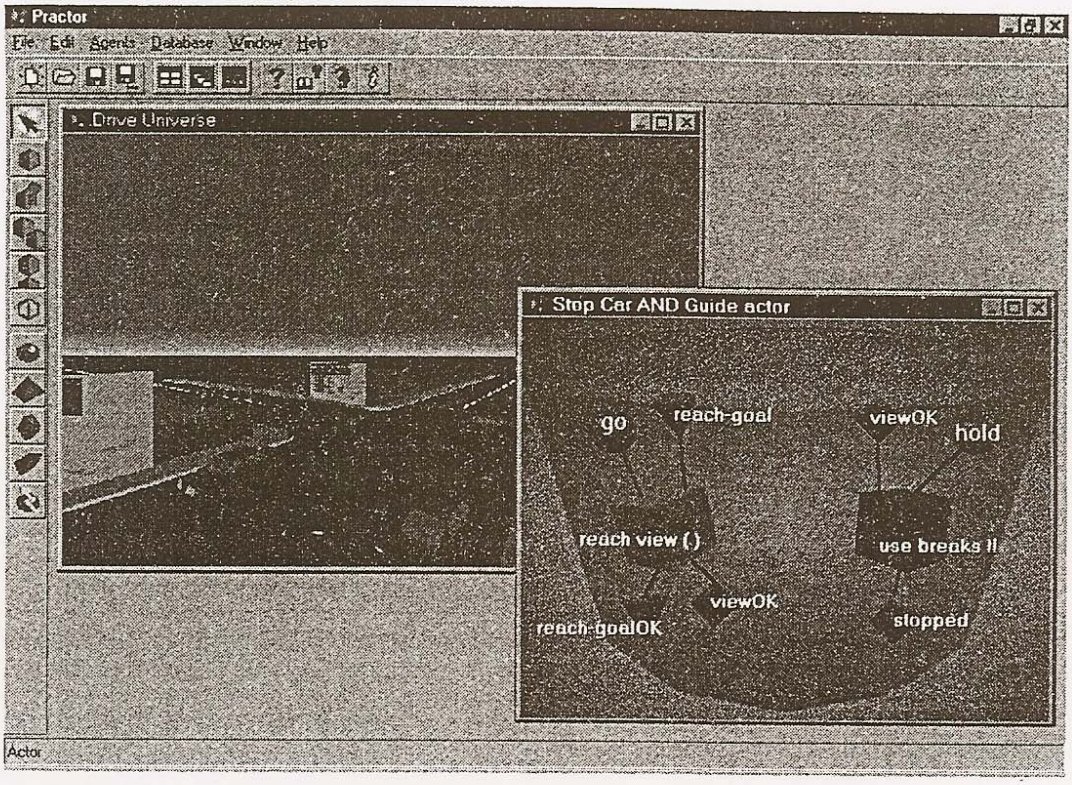

Fig. 3 Snapshot of the common virtual workspace where abstract objects (the components of the VR-MFG) are associated with "actual" objects of the target virtual environment and appropriate agent templates with instructional capabilities are applied.

DrIVE [13] is a virtual training environment intended to train novice car-drivers in common driving situations. It is based on desktop VR, with a minimum of requirements in processing power and data storage. DrIVE is a medium scale application and consists of three main parts: *driving lessons*, *tests for the trainee*, and *free driving* with on-line guidance.

The aim of the application is to transfer experience on the domain of driving behavior, which can be done using the synthetic experience that virtual environments are able to provide rather than actual practicing which involves obvious dangers. DrIVE has been developed with Superscape VRT software. The physical properties were attached to the objects which constitute the DrIVE environment using the appropriate editors. In the first system version the interactive properties were extracted informally and code was assigned incrementally and directly into the objects using SCL (a C-like programming language with event driven code execution) in order to implement these properties. In the second system version, the third part of *free driving* has been designed afresh, using VR-MFG for the interaction design, in the framework of ISW, before any code was assigned to the objects. Then, the implementation of this part was realized, using this formal specification. The benefit was twofold: evaluation of VR-MFG and a substantiated system. Moreover, a

temporal comparison shows that for the implementation of the *free driving part* with the classical method three person-months were needed, and only two person-months using VR-MFG in the interaction specification phase. The design team designed the VR-MFG graphs by hand instead of the 3D Graphical Interaction Editor and typed the intermediate code which constitutes the *Interaction Data Structure* in text files the same way, because the editor and the analysis modules were under development.

The Crossroads example refers to the third part of the application where the user-trainee is the driver of one of the virtual environment's cars. His car reaches the crossroads, and two other cars are coming from the opposite directions.

The user plan-goal decomposition approach will be applied in order to specify the way the user must pass the crossroads safely, which is the main user-goal. This goal is directly decomposed in three sub-goals: *Stop* the car before the crossroads, *Check* and give priorities, and *Pass*.

Two representative graphs will be presented, one for the representation of the overall goal (*Cross-Goal*), and the other for the representation of the *Stop*-subgoal. In Figure 4 the overall goal is decomposed into the three sub-goals, using the *AND guide actor* (represented with a cylinder which includes all the other VR-MFG components).

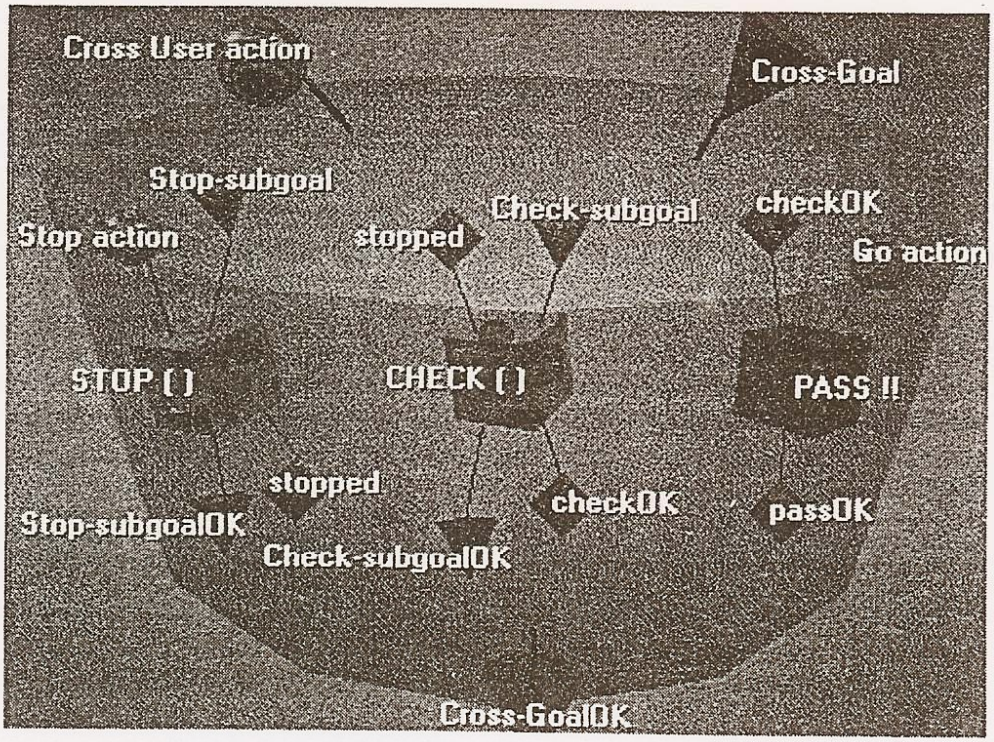

Fig. 4 The *Cross-Goal* represents the user-intention, and the *Cross User action* the actions he must perform to achieve this goal. Inside this *AND Guide actor,* there exist the *STOP ( )* and the *CHECK ( )* context actors, and the *PASS !!* action actor.

The *Stop action* event link (represents the user actions in order to stop the car) and the *Stop-subgoal* context input link (shows that the user intents to stop the car) form the *STOP ( )* context actor set of input links. Also, the *stopped* logical condition link (shows that the car actually stopped) and *Stop-subgoalOK* context link (the existence of a token indicates that the *stop the car* subgoal is being satisfied) form its set of output links. The *stopped* condition is then checked by the *CHECK ( )* context actor which can be further decomposed in order to fulfill the *checkOK* condition, permitting (along with Go action) the *PASS !!* action actor to fire.

In Figure 5 the *STOP ( )* context actor is decomposed. *STOP ( )* context actor includes the *REACH VIEW ( )* context actor and the *USE BRAKES !!* action actor. The user actions in order to drive the car in a position that gives him an appropriate view of the crossroads, are represented by the *Go* event link. The actor *REACH VIEW ( )* fires and a token is produced in the *viewOK* condition. This token is consumed by the *USE BRAKES !!* action actor, if there is a token in the *Hold action* event link, also. Then a token is produced in the *shakeFeedback* perception link (which can be used in order to provide the user with real feedback through an advanced hardware interface device, e.g. a data-glove or a cyber-data-chair), the *stopped* condition link and in the *Stop-subgoalOK* context link. Five more graphs, with almost the same complexity with those presented, are enough in order to complete the entire interaction specification and design for the *Crossroads* example.

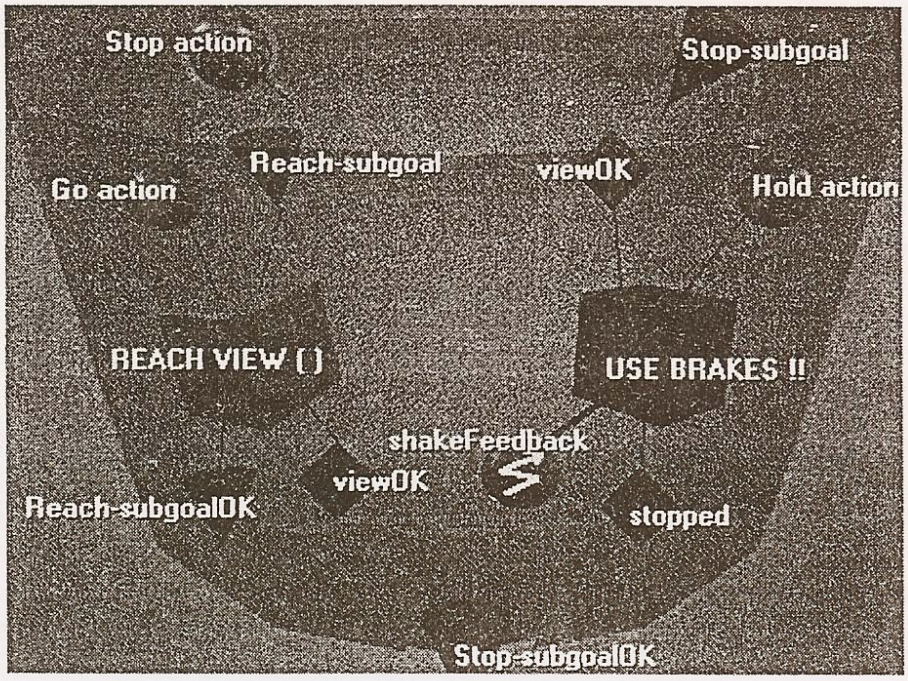

Fig. 5 The *STOP ()* context actor set of input and output links are the same with these presented inside the AND Guide actor of the previous figure. *STOP ( )* includes the *REACH VIEW ( )* context actor and the *USE BRAKES !!* action actor. The condition *viewOK* that exist as an output link of *REACH ( )* actor and as an input link of *USE BRAKES !!* is responsible for the preservation of the correct interaction sequence.

4 Conclusions - Future Directions

In this paper, the Interaction Specification Workspace for the specification and design of virtual environments for training purposes, along with the VR-MFG graphical model and the architecture components were presented. A first working prototype of the system and two training VR applications are currently under development, one of them presented as an example application showing the model capabilities.

ISW architecture provides the author of highly-interactive VR training applications with the capability to specify the interactive dialogue between the trainee and the final virtual training environment. ISW implements the "virtual programming" concept, suggesting the use of virtual environments as 3D workspaces and authoring tools rather than as simple 3D interactive application programs.

Further research will investigate the integration of intelligent agents' behavior models [1, 3, 32] into the basic VR-MFG formalism, and the ISW structure will be extended in the same direction and new guide actors will be added, too. Also, the specification of tutor-learner interaction sessions points to the need of establishing techniques for specifying models of instruction (for example: tutor models the desired task, then student practices on this, teacher coaches him and lets him accomplish the task alone as he gains proficiency). Moreover, network features, formally substantiated will be included [4], providing multi-user (or/and multi-agent) capabilities. The implementation of the architecture of the Prototype Interaction Editor and the Prototyping Mechanism which relies on the model's formalism, will be integrated along with a third-party VR development platform and a prototype system will be constructed. Providing the designer with the capability to switch between the design and prototype process, the complete ISW architecture could serve as the basic platform for the interaction specification and design, for the design of tutorial interactions and for the rapid prototyping of the complete Virtual Training Environments in any application domain.

References

[1] Bates, J. (1994). The role of emotion in believable agents. Communications of the ACM, 37(7):122-125.

[2] Bayarri,S. et al. "Virtual Reality Techniques in urban driving simulation". Proc. Driving Simulation Conference in Real Time Systems '94 pp. 29-43. Paris

[3] Beale, R., Wood, A., "Agent-Based Interaction", in People and Computers IX:Proceedings of HCI '94, Glaskow, UK, August 1994, pp.239-245.

[4] Bell, G., Parisi, A. and Pesce, M., "Virtual Reality Modeling Language: Version 1.0 Specification", May 26, 1995.

[5] Benford, S. et al., "From Rooms to Cyberspace: Models of Interaction in Large Virtual Computer Spaces", in Interacting with Computers (Butterworth-Heinmann), 1993.

[6] Billinghurst, M. & Savage, J. (1996). Adding Intelligence to the Interface. In Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium (pp. 168-176). Piscataway, NJ: IEEE Press.

[7] Bolzoni, M. L. G., "Eliciting a Context for Rules of Interaction: A Taxonomy of Metaphors for Human-Objects Communication in Virtual and Synthetic Environments", Proceedings of the 2nd UK VR-SIG and Contributors, December, 1, 1994, Reading, UK, pp. 78-87.

[8] Bowman, D. A., and Hodges, L. F. "User interface constraints for immersive virtual environment applications", TR GIT-GVU-95-26, Graphics, Visualisation and Usability Center, Georgia Institute of Technology, USA, 1995.

[9] Bricken, M. and Byrne, C. M., Summer Students In Virtual Reality: A Pilot Study On Educational Applications Of Virtual Reality Technology. (unpublished paper) Human Interface Technology Laboratory (HITL) of the Washington Technology Center (WTC) at the University of Washington (UW), 1992.

[10] Burks, L., "Information Architecture: The Representation of Virtual Environments", Harvard University Graduate School of Design, Thesis Document, May 1996.

[11] Card S., K., Robertson, G., G., Mackinley, J., D., Information Visualizer, An Information Workspace, Proceedings of SIGCHI 1991, pp. 181-188.

[12] Deering, M., The HoloScetch VR sketching system, Communications of the ACM, Vol 39, No 5, May 1996, pp.54-61.

[13] Diplas C., Giakovis D., Pintelas P., "DrIVE: A Virtual Training Environment For Driving Behaviour", in Proceedings of the First International Conference on Computers and Advanced Technologies in Education (CATE 96), pp.191-200, March 18-20, 1996, Cairo, Egypt.

[14] Fuchs, J., and Bishop, G., "Research Directions in Virtual Environments. An Invitational Workshop on the Future of Virtual Environments". TR92-027, March 1992, The University of North Carolina at Chapel Hill, Department of Computer Science.

[15] Gobbetti, E., Balaguer, J., F., VB2 An Architecture for Interaction in Synthetic Worlds, Proceedings of UIST'93, November 3-5, Atlanta, Georgia, 1993, pp.167-178.

[16] Harrison, M., D., and Duke, D., J., "A Review of Formalisms for Describing Interactive Behavior", Amodeus Project Document: System Modelling/WP28, January 1994.

[17] Hill, R.W., Johnson, W.L., "Situated Plan Attribution", Journal of Artificial Intelligence in Education, (6)1, pp. 35-67, 1995.

[18] Johnson, W.L., "Pedagogical Agents for Virtual Learning Environments", Proceedings of the International Conference on Computers in Education, pages 41-48, Singapore, 1995.

[19] Kameas, A., "A Formal Model for the Specification of Interaction and the Design of Interactive Applications". Ph.D. Thesis, Department of Computer Engineering, University of Patras, Greece, 1995.

[20] Kameas, A., Diplas, C., Gerogiannis, V. and Pintelas, P., "Encapsulating multiple perspectives in interaction specification". Proceedings of 20th EUROMICRO Conference, Liverpool, England, September 5-10, 1994.

[21] Laurel, B., Computers as Theatre, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1993, p.32.

[22] Mikropoulos, A., Diplas C., Giakovis, D., Halkidis, A., Pintelas, P., "Virtual Reality & Education: New Tool or New Methodology?", Proceedings of 2nd Conference on Informatics in Education, pp.57-67, November, 11-13, 1994, Athens, Hellas.

[23] Opdenbosch, A., and Rodriguez, W., " Interactive Visualizer: Object and View Manipulation Algorithms", Journal of Theoretical Graphics and Computing, STCG, Vol 6 (in press)

[24] Pausch, R., et al, "Disney's Aladdin: First Steps Toward Storytelling in Virtual Reality", Proceedings of Computer Graphics, Annual Conference Series, pp. 193-203, 1996.

[25] Poupyrev, I., Billinghurst, M., Weghorst, S., & Ichikawa, T. (1996). The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR. In Proceedings of UIST '96 (pp. 79-80). New York, NY: ACM

[26] Rickel, J., and Johnson, W., L., "Integrating Pedagogical Capabilities in a Virtual Environment Agent", to be presented in First International Conference on Autonomous Agents, February 1997.

[27] Shaw, C, Green, M. , Liang J. and Sun, Y., Decoupled Simulation in Virtual Reality with the MRToolkit. ACM Trans. On Information Systems (11-3), July 1993, p. 287.

[28] Snowdon, D. N., West, A. J. and Howard T. L. J., "Towards the next generation of Human-Computer Interface", Proceedings of Informatique '93: Interface to Real & Virtual Worlds, 26-26th March 1993, Montpellier, France, pp. 399-408.

[29] Snowdon, D., West, A., "The AVIARY Distibuted Virtual Environment", Proceedings of the 2nd UK VR-SIG and Contributors, December, 1, 1994, Reading, UK, pp. 39-54.

[30] Stoakley, R., Conway, M., J., Pausch, R.. Virtual Reality on a WIM: Interactive Worlds in Miniature. In Proceedings of ACM CHI95, Denver-USA, May 7-11 1995.

[31] Sturman, J. and Zeltzer, D., A Design Method for "Whole-Hand" Human-Computer Interaction. ACM Trans. On Information Systems (11-3), July 1993, pp. 219-238.

[32] Tambe, M., et al, "Intelligent Agents for interactive simulation environments". AI Magazine, 16(1), pp. 15-39, Spring 1995.

[33] VRT 3.60, Superscape Ltd., Reference Manual.

[34] Ware, C., Franck, G., "Viewing a graph in a Virtual Reality Display is Three Times as Good as a 2D Diagram", Proceedings of 1994 IEEE Conference on Visual Languages, S. Louis, Missouri, USA, October, 1994, pp. 182-183.

[35] Whitelock, D., Brna, P., Holland, S., What is the value of virtual reality for conceptual learning? Towards a theoretical framework. In Proceedings of European Conference on AI in Education (in press), 1996, Lisbon, Portugal.

[36] World Up, Sense8 Corporation., User & Reference Manual, 1996.