Toward a Model-Based Approach to the Specification of Virtual Reality Environments*

Daniela Fogli, Piero Mussio
Dipartimento di Elettronica per l'Automazione
Università degli Studi di Brescia
Via Branze 38, 25123 Brescia, Italia
fogli,mussio@ing.unibs.it

Augusto Celentano, Fabio Pittarello
Dipartimento di Informatica
Università Ca' Foscari di Venezia
via Torino 155, 30172 Mestre (VE), Italia
auce,pitt@dsi.unive.it

Abstract

An approach to the specification of a Virtual Reality (VR) interactive environment is presented, which merges and generalizes two methods recently proposed in the literature: the PCL characteristic pattern approach to WIMP system design and the Interaction Locus approach to interactive navigation in 3-D virtual spaces. The merging of the two points of view allows the refinement of the model of interaction of a user with a virtual environment and leads to the definition of "real" and "virtual" characteristic pattern, which the discussion shows to be an important concept for the designer to properly undertake the design of complex virtual reality systems.

1. Introduction

This paper outlines a step toward the definition of a model-based approach for the design of Virtual Reality Environments (VRE). The approach evolves from the *Interaction Locus* proposal for virtual space structuring [15] generalizing the concept of *characteristic pattern* introduced by the Pictorial Computing Laboratory (PCL) [4].

The approach stems from the analysis of the differences between WIMP-based and 3D-based interaction.

At a first glance one could think of extending traditional WIMP-based models to 3D interfaces and to interfaces based on desktop virtual reality in a straightforward way, e.g. by considering the visible projection of the 3D world on the screen as a part of a WIMP-style window, and consequently interpreting the user actions on the screen as actions on the world components. However this view is a purely formal extension which is hardly usable for designing interaction processes and user interfaces behavior. In fact, in a desktop virtual reality environment the translation between the interaction with the screen image and the interaction with the world objects, and the consequent modification of the computation process, are complex and must consider several aspects.

First, the part of the world visible on the screen is only a part of the whole world with which interaction can take place. The user can move in the world by revealing and hiding portions of the world (i.e., bringing in and out from the screen the 2D projections of the 3D world views). This is similar to what happens in a WIMP interface when the user moves a window partly off screen, but on a bigger scale. The 3D world is normally a large environment in which movement (therefore selection of the visible scene) is the primary interaction method, while in WIMP interfaces desktop rearrangement is occasional. In terms of interaction design this means that the identification of the area subject to user interaction is more critical in a VR scenario.

Second, the desktop is a 2D projection of a 3D scene. The position of the user pointing device with respect to the objects on the screen is not defined by the screen layout, but requires additional information about the user distance from the elements depicted (e.g., in terms of z-axis coordinates). As a consequence, a click on an object image does not correspond necessarily to a "click" (i.e., a selection) on the object, and the same user action has different interpretations depending on position parameters not immediately perceivable. The current state of the computational process must take account of it, but this state is completely hidden in the interface look. We do not address here psychological issues related to user perception of position in virtual spaces. It is however evident that the lack of information about accessibility of an object in the virtual space impacts the interaction model.

We address these problems generalizing the PCL model of Human Computer Interaction (HCI) to account for VR entities. The characterizing features of this model leads to the definition of an *Interaction Modeling Space* in which each space dimension represents types of lan-

* A short version of a part of this paper has been presented at HCC02, 2002 IEEE Symposia on Human-Centric Computing Languages and Environments, Arlington, VA, September 3-6, 2002.

guages: 1) *programming languages* to specify system computations; 2) *user activity languages* to specify user activities; 3) *characteristic structure languages*, which are languages devoted to deal with the physical characteristics of the messages from the machine to the user, in analogy with the languages of sculptures and paintings introduced in a seminal work by Stiny and Gips [18].

The HCI model and the Interaction Modeling Space constitute the frame in which a Virtual Reality Environment is specified. In a first step, the requirements to be satisfied by the VRE are identified using the HCI model. Then the VRE is characterized in the Interaction Modeling Space, so obtaining the specification of VRE computational and interaction characteristics.

This paper is organized as follows: Section 2 gives an overview of the literature relevant for our work. In Section 3 the PCL model for WIMP systems and the Interaction Locus concept are briefly reviewed. Section 4 revises the Interaction Modelling Space introduced in [2]. Section 5 discusses a possible extension of the HCI model to Virtual Reality. Section 6 presents a formalization of this extension based on the Interaction Modelling Space revised. Section 7 draws the conclusion and suggests further development.

2. Related work

The model and the approach to VRE specification discussed in this paper was incrementally defined abstracting and generalizing several experimental cases and was influenced and is related to several methods and results documented in the literature. The model is a refinement of the Norman's cyclic model [10]. In fact, the Norman model focus on the user's view of the interaction, and does not attempt to deal with the system communication through the interface and its influence on system organization. Black box models – for example the PIE one [8], first address the communicational aspects. The VRE approach starts from the PCL model [3] and takes care both of the user and of the machine being *holistic* and *syndetic*. It is holistic – in the sense of Preece et alt., [17] – in that "the decision about the way an interface should look are made in relation to how this will be physically communicated to the users". However, contrary to most holistic approaches, the PCL formalizes the description of the interaction process, allowing a designer to formally specify [4] his/her own conceptual model intended as a technically accurate model of the computer system created by designers for their specific purposes.

In this way, the formalism responds to a requirement often advanced in the literature [7, 9, 13] that a specification has to take into account not only the computational constructs (organization of data and programs) but also explicitly address the organization of information on the surface of the system.

The VRE design approach is syndetic [1] in that the model of HCI interaction is framed in a "macro-theory" in which a human – a psychological system- interacts with a computational system, the VRE. The VRE "micro-theory" is developed within the frame of the Interaction Modeling Space, and linked to the Human micro-theory, by the HCI model. The Interaction Modeling Space is developed in analogy with the multilevel machine of Tanenbaum [19], taking however into account the two new dimensions necessary to characterize the system interactive behavior.

For what concerns 3D environments, there have been very few efforts for the formalization of interactive scenes. One of the most significant approaches has been proposed by E. Jacob et al. [11, 12]; their work includes a preliminary analysis that emphasizes how at present there are a number of good specifications for current direct manipulation WIMP interfaces, but a substantial lack of formalization for 3D worlds.

The authors propose, as a solution to this scarcity, a language that specifies all the aspects of the interaction; their model and language map closely to the user's view of the fine-grained interaction in a non-WIMP interface. Their approach has two interesting points: the ability to specify the user input, a feature missing in other interaction languages like VRML, and the ability to describe both non-WIMP and WIMP interfaces; this can be extremely useful with desktop virtual reality, where we can have a mix of three-dimensional and bi-dimensional widgets to control interaction.

In spite of these positive features, the proposed formalization still misses support for authors, in terms of high level vision and reusability.

For what concerns the problem of navigation, one of the main interaction tasks in 3D virtual worlds, an interesting approach has been advanced by R.P. Darken and J.L. Silbert [6]; they suggest the need of a wise design of the structure of the environment and of navigational helps to improve navigation. Their work stems from a critic analysis of the navigation problem in the real word, leading the authors to the conclusion that in many cases design principles from the reality can be applied to the virtual worlds.

3. Modelling interaction in WIMP and VR interfaces

3.1. The PCL Human Computer Interaction Model

The Pictorial Computing Laboratory approach models the HCI process as a sequence of cycles: the human detects a set of events generated by the machine – the image on the screen, a sound from the microphone, etc. –, derives their meaning, decides what to do next, manifests his/her intention by an activity performed operating on the input devices of the system; the system perceives these operations as a stream of input events, interprets them, computes the response to human activity and materializes the results through its output devices, so that they can be perceived and interpreted by the human. In principle, this cycle is repeated until the human decides that the process has to be finished, because the task has been achieved or failed.

Let us restrict to WIMP interaction, which was studied by the PCL in details [3]. On one side, in each cycle, the human interprets the image on the screen by recognizing *characteristic structures* (cs), i.e., sets of image *pixels* which he or she perceives as functional or perceptual units. The css recognition and interpretation results into the association of a meaning with a structure. Identification and interpretation (or misinterpretation) of css are influenced by the similarity (dissimilarities) with real tools and graphical constructs traditionally adopted by the user community to perform and document their activity. The human deduces the meaning associated with the whole image by combining the meanings of the css recognized in it.

On the other side, in each cycle, the system plays several roles. First, it is the medium conveying the messages on which interaction is based: a set Pi of hardware tools and hardware implemented algorithms maintain and display the images on the screen. Second, the system plays the role of the second interacting entity, as a set of *application programs* (Ap): some of its sub-programs compute the system reaction to the user activity. The input to Ap are computed by a set of devices and programs which capture and digitize the input operations performed by the human, relate them to the image on the screen and assign them a meaning: this third set of devices and programs plays the role of the *tools* used by the human to manifest his/her intention. Last, there are some programs which send to Pi the results of the computation performed by Ap, determining the image on the screen, acting as the *materialization tools* for Ap. From the designer point of view, the system is a modular entity, arising from the composition of several sub-systems.

To specify the system structure and behavior, the concept of *virtual entity* (ve) is introduced. Virtual entities are computational entities which manifest their appearance allowing a user to interact with an application program, i.e., they receive user inputs and convey messages from the application program to the user.

A virtual entity ve is generated by a software system which organizes four (sets of) programs: the programs Pi acting as image support, the program Ap acting as a second communicant, and programs FI and FO acting as I/O tools. A ve is therefore specified as ve = <Pi, Ap, <FI,FO>>.

3.2. The Interaction Locus model

The *Interaction Locus* concept was introduced in the context of a research that aimed at finding weaknesses in the current interaction modalities in 3D environments such as virtual worlds. Navigation is a primary task in interacting with virtual environments: it is often the main activity performed by the user and, due to the explorative approach that characterizes interaction in virtual environments, it is a prerequisite for more sophisticated behaviors. In spite of that current 3D interfaces don't offer satisfactory solutions to support this task.

In order to increase the quality of interaction in virtual worlds a better formalization of the navigation problem was given in [16], distinguishing three related primary issues: identification of the scene structure, orientation and navigation. A number of guidelines for virtual worlds were critically extrapolated from the analysis of the real world and applied to virtual environments; these guidelines included also a strong attention toward a parallel aspect characterizing the real world: multimodality, which means giving and receiving different information simultaneously from different senses [14]. The concept of *Interaction Locus* (IL) incorporates these guidelines, helping the user to identify the scene structure using multimodal communication channels, e.g., visual, auditory, tactile, hypertextual, etc.

But the introduction of the Interaction Locus concept has not only navigational benefits. In a *well defined* 3D environment there should be no dichotomy between morphology and function, and in many situations there is a coincidence between the division of spaces and activities performed in them. This is not a casualty, but it is generally the result of a designer's work; in that situations the morphology of space has a sense in itself but also it helps to identify areas characterized by precise functions.

The Interaction Locus concept incorporates this twofold morphologic-functional vision. Therefore in a well defined 3D environment we can associate to the Interaction Locus not only a tuple of coordinated multimodal messages that informs the user about the nature of that portion of space, but also a set of possible interaction activities.

If we consider the authoring of 3D experiences, with the Interaction Locus concept the author is enabled to build virtual interactive experiences evidencing the areas that are characterized by coherent morphologic features and by homogeneous interaction modalities; i.e., the author is enabled to superimpose to the 3D scene a virtual entity whose task is to inform the user about the nature of the part of the scene he/she's entered and to present and to mediate the possible interactions inside the area controlled by it.

3.3. Specifying the state of the interaction process and its dynamics

Given a ve = <Pi, Ap, <FI, FO>>, its current state is called *characteristic pattern* (cp), defined as a triple cp = <cs, u, <int,mat>> linking the current state u of the program Ap to the digital events perceivable by the user and generated by the computation. The set of digital events is the generated *characteristic structure* cs: in the WIMP case it is the set of pixels visible on the screen; in a virtual reality space it is the set of voxels which shape a 3-D object; in a multimodal environment it is the set of perceivable elements generated by a ve having a specific and recognizable function during the interaction. The functions int (interpretation) and mat (materialization) describe the relations of components of the cs with components of u as computed by FI, and the relations of components of u with components of the cs as computed by FO.

The state of the overall process is described as a triple vs = <i,u,<int,mat>>, where i is the array of pixels constituting the current image, u is a suitable description of the current state of the process Ap, int and mat define the relations of elements of i with components of u. This triple is called *visual sentence* (vs). According to this definition, a visual sentence is a special cp whose cs is the whole image on the screen, i.e. its image part i is the current message to be interpreted by the human and by the system. A vs describes the state of the computation performed by the set of programs which constitutes the VRE.

The characteristic pattern describes the state of the computation performed by (sub)programs constituting a ve. The dynamics of the interaction are described introducing the concept of *transformation*, which links the activities performed by the user and recognized by the system to the computation under execution.

The environment with which a user interacts is seen as a *virtual environment* in which a population of virtual entities ve is present, and which can be described specifying the behavior of the population. The dynamics of a ve during the interaction process is specified as a rewriting system, whose rules describe how the current cp (i.e., the current state of a ve) evolves in reaction to the user activities on the system.

In each interaction cycle, a visual sentence vs_1 is transformed into a visual sentence vs_2 as the consequence of some human activity a. In a WIMP system, the human performs an activity operating on an input device – say clicking a mouse button – in relation to some cs recognized on the screen. The system relates the operation to the current active cs (in WIMP systems, the one pointed to by the mouse pointer) and interprets it as a command from the user. Then it fires the consequent computation, referred to in the u associated to the cs in the corresponding cp, which often implies the change of the appearance of the cs.

The designer describes the human activity as a pair a = <operation, cp> and specifies the transformation as tr = <a,<vs_1,vs_2>>, where cp in a is present in vs_1. The interaction process is specified as a sequence of such transformations. In a transformation, vs_1 and vs_2 share a common part, while the variable part of vs_1 is transformed into the variable part of vs_2 through the application of a transformation rule in the form tr = <a_i, r>, where $a_i \in A$ is the user activity and r is a rewriting rule. A rewriting rule is a triple r = <ant, cond, cons>, where ant (antecedent) is a set of cps, cond is a condition on ant, and cons (consequent) is a second set of cps. A rule is applied transforming vs_1 into vs_2 if ant appears in vs_1 and cond is satisfied. Let TR be a set of transformation rules [5]. The possibly infinite set of all the sequences of vss that, starting from the initial vs_0, are determined by all the sequences of executable user actions and system computations is a visual language called Interactive Visual Language (IVL). Each sequence in IVL describes a specific user-computer interaction session. On the whole, IVL is specified as IVL= <vs_0, TR>, from which every sequence of admissible transformations can be computed.

With reference to the Interaction Locus approach, the interaction can be described in terms of the HCI model introduced above: each virtual entity ve present in the environment can be enriched by the capability of manifesting information which help users in understanding the characteristics of the ve and how to interact with it. Therefore the interaction locus is viewed as a portion of environment in which virtual entities deputed to interaction exist and are perceivable to the user supporting him/her during exploration.

The PCL approach can be extended to virtual reality spaces modelled by the interaction locus approach, by defining a hierarchy of characteristic structures cs and computation processes u which allow a designer to split the complexity of the interaction design in levels, starting from a very abstract one, in which the three components of the cp and user activity are defined using high level metaphors, down to the elementary digital events, computational elementary steps and elementary user activities and the metaphors for interaction defined in desktop virtual reality interfaces.

Moreover, the merging of characteristic pattern and interaction locus approaches allows the distinction between “real” and “virtual” characteristic patterns, which is an important concept for the designer to properly undertake the design of complex virtual reality systems.

4. Three language dimensions: pictorial, programming and user activity

The interaction process was modeled in [2] by the introduction of an *Interaction Modeling Space*. In this space a partially ordered set of interaction machines is represented, in analogy with the hierarchy of real and virtual computing machines used in operating systems and discussed for example in [19]. In the classical OS approach, real and virtual machines were defined through the languages in which the algorithms of interest are described. These languages are defined at different levels of abstraction, from the machine language level to the high level languages. In modeling ve the computational process can be specified in analogy, describing a program Ap, whose interpretation determines the sequence of states of computation u. However, to define the ve with reference to the interaction process, it is also necessary to define a) the user activities a as perceived by the machine, which constitute the inputs determining the development of the computation in time; and b) the output of the ve dynamics, i.e. the cs which allows the user to decide which activity to perform.

The set of user activities as well as the set of css can be described starting from a set of atomic elements to form more complex ones. For this reason we speak of activity languages and of perceptual languages of css. As for programs, also activities and css can be described at different levels of abstraction, by the definition of low level – close to the machine events – or high level – user and task-oriented languages –. Accordingly, we can define concrete and abstract interpreters of activities and css. The combination of program, activity and cs interpreters into one abstract or concrete machine is a ve generator, capable of interpreting the user actions and generating css. In other words, it is an Interactive Machine, which can be a Virtual Interactive Machine (VIM) or the Real Interactive Machine (RIM).

Figure 1 shows a generalized version of the Interaction Modelling Space presented in [2]. On each axis, close to the origin, languages at the lowest level of abstraction are put. Each point in this space represents a set of VIMs, i.e. a set of hypothetical interactive visual systems defined by their user activity language, their programming language and their characteristic structure language. Each point is positioned in the space according to the processes of abstraction which have been performed along the three axes to create it. The VIMs in the set are different instances of the same functional specifications implementation at that level of abstraction.

This space is here extended to a general case of ves which manifest their existence not only generating a 2-D image but also sound and which are defined in a 3-D space. The characteristic structure of the ve becomes in this case the current visual appearance and other multi-

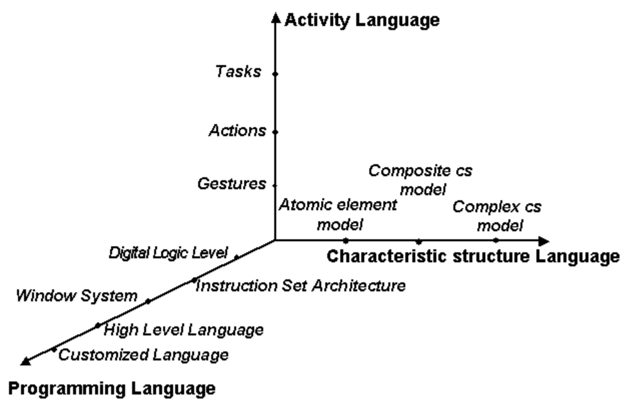

Figure 1. The Interaction Modelling Space

modal manifestations of the 3-D ve, which are now the elements of a Characteristic Structure Language.

The sets of virtual machines in the space are partially ordered by translation relations among interpreters on each axis. This set admits a minimal, i.e. a set of IMs generating a ve which are less abstract than any other along the three axis. These IMs are the Real Interactive Machines, from which the others are derived by abstraction processes applied to one or more axis.

5. From 3D models to characteristic patterns

In 3D virtual environments, we observe that the correspondence between a characteristic structure defined as a portion of screen image (a set of pixels) and the ve it represents is not direct, but mediated through a chain of translations. Let us illustrate this concept with an example.

A virtual world is made of ves some of which provide information, some are interaction elements (in a wide sense, they are sensors of the user actions) and some others are *passive* elements which have been put on the 3D world to give a structure to the scene or to enrich it providing a sense of realism. The scene can be described as a set of ves having some properties, rendered according to a metaphor, a set of geometries, and a set of drawing details, assuming the definition of a virtual interaction machine in which activities, css and programs are defined at the desired level of abstraction. We limit our example to a basic sequence of interaction steps made of a selection of a ve of the world, and the modification of such ve (e.g. a change in the shape) showing a visible feedback of the user action. The screen layout is defined and generated by a sequence of translations:

1. the ves are built as 3D shapes (simple or complex), having physical properties in terms of size, position in a reference space, color, texture, and other optional

features (e.g. transparency, softness, etc.) with associated computational properties.
2. The 2D css are rendered with appropriate colors and surface textures properties by projecting the scene from the specific user point of view onto the screen plane.
3. The projected scene is clipped by the size of the user window.

Step 1 is executed at the beginning of the virtual world instantiation. Steps 2 and 3 are repeatedly executed as the user moves or changes her/his orientation in space. User interaction is handled in the same way:

a) The user selects a cs on the screen, the projection of the 3D virtual shape of the ve on the screen. The interpreter verifies the user z-axis coordinate in order to check if the selection occurs on a ve or not. If not, there is really no interaction and the scene does not change.
b) If yes, the ve is identified, and a message is sent to the 3D ve in the virtual world model.
c) The behaviors associated to the ve are checked, and the ve modified accordingly. This generates a change in the world, therefore steps 1-3 above are executed with the new world configuration. Figure 2 summarizes this process.

The dynamics of ve under selection is specified by a a set TR3D of transformations linking user activities to rewriting rules. The rules describe the dynamics of the ve as 3D object and are applied in the 3D virtual world. Note that the cp = <cs,u, <int, mat>> now links a 2D cs – what is perceivable by the user on the screen – to a computational construct u which describe the state of a 3D ve. The programs implementing int and mat must verify where the activity of the user occurs in the 3D space as well as are in charge of projecting the 3D shape of ve onto the cs taking into account the user point of observation and vice versa.

Hence in a rewriting rule the condition cond requires that the user activity occurs on the ve in the 3D space to be satisfied.

With a reference to the PCL model the above process could be described in only one step by assuming that the int and mat functions corresponding to the current visual sentence take care of the different levels of computations needed. Indeed, this is what is perceivable by the user, who clicks and sees the ve changing its shape, i.e., from the state cp_1 = <cs_1,u_1,<int,mat>> the systems goes to the state cp_2 = <cs_2,u_2,<int,mat>>, where:

- cs_1 is the area of the screen related to the selected ve before the click, cs_2 the same portion after the ve shape modification;
- u_1 is the state of the computational process before the ve selection, u_2 the state after the ve selection and modification are completed;

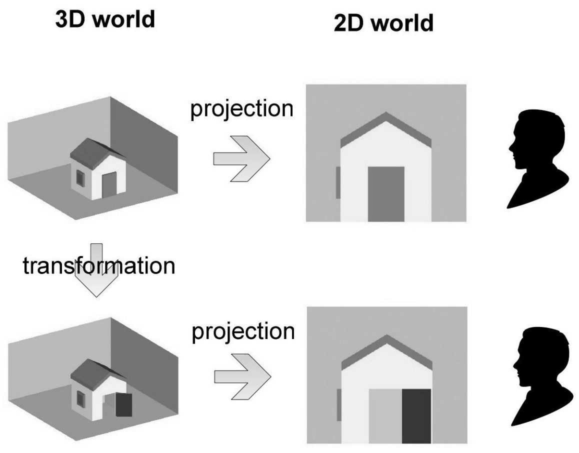

Figure 2. Interaction transformation between 2D and 3D

- int and mat are the two functions which, through different levels of data transformation, carry on the computation process.

This view, illustrated in Figure 2, does not make evident the different steps involved, which can be made explicit through a generalization of the concepts of characteristic structure, characteristic pattern and interaction process.

6. Managing the abstraction levels in VRE design

To implement a system starting from an abstract definition of VIM at level *(i,j,k)*, i.e. belonging to VIM_{ijk}, it is necessary to express the activities, the computational states, and the css of the abstract machine in terms of the execution performed on a real machine. The point (0,0,0) in the Interaction Modelling Space identifies a set of computationally equivalent Real Interaction Machines, in which each program, activity and cs is considered at the lowest concrete level of abstraction. u_0, a_0 and cs_0 are directly interpreted by the real machine, and a high level specification in VIM_{ijk} can be translated into a Real Interaction Machine RIM in VIM_{000}, through a set of transformations which define the VIM behavior in terms of RIM.

A transformation is defined by a characteristic structure cs_i, an action a_j and a state of program u_k. Therefore:

- cs_i, defined at level *i*, must be translated to level 0 (cs_0)
- a_j, defined at level *j*, must be decomposed to level 0 (a_0)
- u_k,, defined at level *k*, must be interpreted to level 0 (u_0)

- the int function is defined as $\mathsf{int}_{i,k}$, mapping cs_i to u_k, and must be decomposed to operate on cs_0 and u_0
- similarly, $\mathsf{mat}_{i,k}$ is defined on cs_i and u_k, and must be decomposed to operate on cs_0 and u_0.

The definition of cp as the current state of a virtual entity is related to the actual visible behavior of the computing system. We use the term "*real* cp" to denote this process. For each real cp, a topological structure (a partially ordered set) of *virtual cps* is defined, where a cp specification at one level of the topology offers some utilities to the higher level and hides the details of the cp specifications at lower levels, as much as in the partially ordered set of virtual machines.

In this structure, a lowest level is always present, denoted here by $\mathsf{cp}_{0,0} = <\mathsf{cs}_0, \mathsf{u}_0, <\mathsf{int}_{0,0}, \mathsf{mat}_{0,0}>>$. Higher level specifications of virtual cps are not totally ordered, since they can belong to different abstraction kinds, i.e., different metaphors. An order relation can be established between the components of different cps, i.e. between pairs of *virtual* css[1], of states of the process u, and of int and mat functions.

This can be formalized by specifying, for each component in a $\mathsf{cp}_{i,k} = <\mathsf{cs}_i, \mathsf{u}_k, <\mathsf{int}_{i,k}, \mathsf{mat}_{i,k}>>$ in the structure, a process of *translation*, or *interpretation*, or *combination* from one level to the lower one.

A *translation process* maps one (virtual) cs into the cs at the lower abstraction level. For example, at a high level, the cs of a 3D ve can be regarded as a connected 3D space, which is translated at the lower level into a cs constituted by the set of points in the 3D space identifying the ve. Going to a lower level again, this cs can be translated into a cs constituted by the 2D points representing the projection from 3D to 2D of the ve. More in general, let us call g the translation function which allows the translation of a characteristic structure cs_{i+1} at a higher level into a characteristic structure at a lower level cs_i, $\mathsf{cs}_i = \mathsf{g}_i(\mathsf{cs}_{i+1})$, where g_i is the translation function of the *i*-th level.

An *interpretation process* is carried out, at a low level, to interpret the computational construct u_k of a higher level $\mathsf{cp}_{i,k}$. Let us call f_k the conversion function which allows the conversion of a computational construct u_{k+1} at a higher level by a computational construct at a lower level u_k. Therefore, u_k is equal to $\mathsf{f}_k(\mathsf{u}_{k+1})$.

At the lowest level, the virtual characteristic pattern $\mathsf{cp}_{0,0}$ corresponds to the real characteristic pattern as defined above, that is to the state of a ve as generated by a computational process at a given instant.

In particular, the characteristic structure cs_0 is obtained by chain of translations through the application of functions $\mathsf{g}_0, \mathsf{g}_1, ..., \mathsf{g}_{n-1}$, i.e.,

$$\mathsf{cs} = \mathsf{cs}_0 = \mathsf{g}_0(\mathsf{g}_1(...\mathsf{g}_{n-1}(\mathsf{cs}_n))).$$

Similarly, the state of the computational construct u_0 is obtained by a chain of conversions through the application of functions $\mathsf{f}_0, \mathsf{f}_1, ..., \mathsf{f}_{m-1}$, i.e.,

$$\mathsf{u} = \mathsf{u}_0 = \mathsf{f}_0(\mathsf{f}_1(...\mathsf{f}_{n-1}(\mathsf{u}_m))).$$

Finally, a *combination* process is used for int and mat functions to implement the interpretation of user messages and materialization of system messages both at a low level. The interpretation and materialization function at the level (0,0) are obtained by the combinations of the translation and conversion functions in the two processes.

On the activity dimension, activities defined at a high level of abstraction a_{i+1} is mapped into lower level activities by a function h_j, $\mathsf{a}_j = \mathsf{h}_j(\mathsf{a}_{j+1})$.

The families of functions f, g and h are thus used to specify how a VIM at a certain level can be translated into a machine of different level.

7. Conclusion

In this paper we have recalled the PCL Interactive Visual Language model based on the concept of *characteristic pattern*, which relates the visible properties of the interaction entities to the state of the computational process through a pair of interpretation and materialization functions. The interaction between a user and a machine can be described as a sequence of cp transformations. We have then considered the interaction paradigm in a desktop virtual reality environment based on the concept of *Interaction Locus*, a coherent set of morphologic features and homogeneous interaction modalities for exploration of virtual worlds.

We have then extended the PCL model to interaction in a virtual world by introducing a set of characteristic pattern definitions, able to describe the relationship between the user perception, activities and computation processes at different levels of abstractions. Translations between different levels can map abstract interaction with 3D world objects onto their 2D representation on the user screen. Conversely, the user perceives modifications in the 3D world through a mapping of the scene on the screen.

The presence of different levels of virtual characteristic patterns help the designer to model the interaction in a complex environment such as a virtual world by separating the computation related to the interpretation of the user tasks from the computation needed to interpret, at low level, the user gestures. It can also support the design

[1] The characteristic structure according to the PCL model is the perceivable aspect of a virtual entity, therefore the cs of the more abstract levels have to be regarded as virtual.

of re-usable interaction experiences by translating the same behavior at a high level of abstraction into different concrete implementations, e.g., desktop VR at one side and immersive VR at the other side.

While most of the ideas have been discussed with a reference to the visual interaction, in principle also other interaction modalities can be grounded on the same model. Further work will be directed to investigate specific properties of audio and tactile interaction, possibly extending the concept of characteristic structure to generic multimodal perception of information and interaction entities.

8. Acknowledgments

This work is supported by Italian Ministry of University and Research, MIUR, in the framework of the project *Specification, Design and Development of Visual Interactive Systems, PRIN 2000.*

9. References

[1] P. Barnard, J. May, D. Duke, D. Duce, "Systems, Interactions and Macrotheory", *ACM Trans on HCI*, 7(2), 222-262, 2000.

[2] P. Bottoni, M. F. Costabile, D. Fogli, S. Levialdi, P. Mussio, "Multilevel Modelling and Design of Visual Interactive Systems", in *Proceedings IEEE Symp on Human-Centric Computing Languages and Environments*, Stresa, Italy, 256-263, 2001.

[3] P. Bottoni, M. F. Costabile, S. Levialdi, P. Mussio, "Defining Visual Languages for Interactive Computing", *IEEE Trans. on SMC*, 27(6), 773-783, 1997.

[4] P. Bottoni, M. F. Costabile, P. Mussio, "Specification and Dialog Control of Visual Interaction", *ACM Trans. on Programming Languages and Systems*, 21(6), 1077-1136, 1999.

[5] P. Carrara, D. Fogli, G. Fresta, P. Mussio, "Toward overcoming culture, skill and situation hurdles in human-computer interaction". To appear in *International Journal Universal Access in Information Society*, 2002.

[6] R. P. Darken and J. L. Sibert, "Wayfinding strategies and behaviors in large virtual worlds", in *Proceedings of CHI'96*, ACM, pp. 142-149, 1996.

[7] G. de Haan, "ETAG. A formal model of Competence Knowledge for user interface design", SIKS Dissertation No. 2000-4, 2000

[8] A. Dix, J. Finlay, G. Abowd, R. Beale, *Human Computer Interaction*, Prentice Hall, London, 1998.

[9] T.R.G. Green, F. Schiele, S.J. Paine, "Formalization Models of User Knowledge in HCI", in C.G. van DerVeer et al. (eds), *Working with Computers: theory versus outcome*, Academic Press, 3-46. 1988.

[10] E.L. Hutchins, J.D. Hollan, and D.A. Norman, "Direct Manipulation Interfaces", in D.A. Norman and S.W. Draper (eds), *User Centered System Design: New Perspectives on Human-computer Interaction*, 87-124, Lawrence Erlbaum, Hillsdale, N.J., 1986

[11] R.J.K. Jacob, "A specification language for direct-manipulation user interfaces", *ACM Transactions on Graphics*, 5(4), 283-317, 1986.

[12] R.J.K. Jacob, L. Deligiannidis, S. Morrison, "A software model and specification language for non-wimp user interfaces", *ACM Transactions on Computer-Human Interaction*, 6(1), 1-46, 1999.

[13] T. P. Moran, "The Command Language Grammar: a representation for the user-interface of interactive systems", *Int. Journal of Man- Machine Studies*, 15(1), 3-50, 1981.

[14] D.A. Norman, *Things that make us smart - Defending human attributes in the age of the machine*. Addison-Wesley, 1993.

[15] F. Pittarello, A. Celentano, "Interaction locus: a multimodal approach for the structuring of virtual spaces", in *HCITALY 2001, Human-Computer Interaction Symposium*, Florence, Italy, September 2001.

[16] F. Pittarello, "Multi sensory guided tours for cultural heritage: the Palazzo Grassi experience", *International Cultural Heritage Informatics Meeting (ICHIM)*, Milan, September 2001.

[17] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, T. Carey, *Human-Computer Interaction*. Addison-Wesley, 1994.

[18] G. Stiny, J. Gips, "Shape Grammars and the Generative Specification of Paintings and Sculptures", *Proc. IFIP Congress'71*, 1971.

[19] A. S. Tanenbaum, *Structured Computer Organization*, Prentice Hall, Upper Saddle River, 1999