Virtual Reality Interaction: the Characteristic Pattern Approach*

A. Celentano[1], D. Fogli[2], P. Mussio[2], F. Pittarello[1]
[1] *Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy*
[2] *Dipartimento di Elettronica per l'Automazione, Università degli Studi di Brescia, Italy*
auce@dsi.unive.it, fogli@ing.unibs.it, mussio@ing.unibs.it, pitt@dsi.unive.it

Abstract

We merge the Pictorial Computing Laboratory (PCL) approach to WIMP interaction with the Interaction Locus approach to structuring visual spaces as a step toward the definition of a rational methodology for the design of Virtual Reality interactive systems. 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.

1. Modeling the HCI process

In this work we merge the Pictorial Computing Laboratory (PCL) approach to WIMP interaction [1] with the Interaction Locus approach to structuring virtual spaces [3] as a step toward the definition of a rational methodology for the design of Virtual Reality (VR) interactive systems.

The *PCL approach* models human-computer interaction (HCI) as a process in which the user and the computer communicate by materializing and interpreting a sequence of messages at successive instants of time. Each message is constituted by a set of perceivable elements, called *characteristic structures*. Two interpretations of each message arise in the interaction: one performed by the user achieving the task, depending on his/her role in the task, as well as on his/her culture, experience, and skills, and the second internal to the system, associating the message with a computational meaning, as determined by the programs implemented in the system [2]. The extension of this approach to desktop virtual reality requires not only an accurate definition of the computational processes and a refinement of its specification, but also a better definition of the machine generating them.

The *Interaction Locus* is the "basic element to give a structure to 3D space, as a *summa* of coordinated 3D representation, auditory signs and hypertextual information" [3], i.e., a coherent integration of multisensorial information and multimodal interaction. It was introduced in the context of a research that aimed at overcoming weaknesses in the current interaction modalities in 3D environments such as virtual worlds [4]. 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. This goal is achieved by superimposing to the *raw* 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 its area.

Generalizing the PCL model, the environment with which a user interacts is seen as a *virtual space* in which a population of virtual entities (ve) is present, and which can be described specifying the behavior of the population. The state of each virtual entity is formalized as a *characteristic pattern* (cp), defined as a triple <cs, u, <int, mat>> linking the current state u of the computation generating the virtual entity to the digital events perceivable by the user and generated by the computation. The set of digital events is called *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 and vice versa.

The dynamics of a ve during the interaction process is specified as a *rewriting system*, whose rules describe how the current cp (the current state of a ve) evolves in reaction to the user activities on the system.

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. Such merging defines a hierarchy of characteristic structures cs and computation processes u which allow a designer to split the complexity of the interaction design in levels. They range 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.

2. The interaction model revised

The HCI process is viewed as a sequence of cycles: the human detects the image on the screen, derives the message meaning, decides what to do next, and manifests his/her intention by 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 on the screen, so that

* Work supported by MIUR, Italian Ministry of Education, University and Research, in the framework of PRIN 2000 program.

they can be perceived and interpreted by the human.

This process is formalized by a set of consistent rewriting rules. These rules are defined in terms of sets of user activities, cs on which these activities are performed and programs defining the system reactions. A Visual Interactive Machine (VIM) is represented in an *Interaction Modeling Space* [2], in which each space dimension represents types of languages: 1) programming languages to specify system computations; 2) user activity languages to specify user activities; 3) pictorial languages to specify the images appearing on the computer screen, whenever only visual messages are considered to be exchanged between the human and the computer. On each axis, languages at the lowest level of abstraction are put close to the origin. Each point in this space represents a VIM, and is positioned according to the processes of abstraction which have been performed along the three axes to define it.

This space is extended to a general case of ves which manifest their existence generating not only 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 appearance of the 3-D ve and the sound currently emanating from it, which are now the elements of a *Characteristic Structure Language*. Characteristic Structure Languages are in general languages devoted to deal with the physical characteristic of the messages from the machine to the user [5].

3. Virtual Reality interfaces and interaction

The extension of the above model to 3D interfaces and to interfaces based on desktop Virtual Reality could be done straightforwardly by considering the visible projection of the 3D world on the screen, and by consequently interpreting the user actions on the screen portions corresponding to the world ves. However, in a desktop VR environment the translation between the interaction with the screen image and the modification of the computation process must consider several critical aspects. E.g., the part of the world visible on the screen is only a part of the whole world with which interaction can take place. 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 rearrangement of the desktop 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.

The desktop is a 2D projection of a 3D scene. The position of the user pointing device with respect to the ves on the screen is not defined in the screen layout, but requires additional information about the user distance from the ves depicted (e.g., in terms of z-axis coordinates). As a consequence, a click on a ve image does not correspond necessarily to a "click" (i.e., a selection) on the ve, and a 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.

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 user actions) and some others are purely "decorative" elements which enrich the scene without providing other than 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 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 user window size.

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. The user interaction is handled in the same way:

a) The user selects a cs on the screen, the projection 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.

The dynamics of ve under selection is specified by a set 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 describes 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.

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 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 cp1=<cs1,u1,<int,mat>> the systems goes to the state cp2=<cs2,u2,<int,mat>>, where:

- cs1 is the area of the screen related to the selected ve before the click, cs2 the same portion after the ve shape modification;
- u1 is the state of the computational process before the ve selection, u2 the state after the ve selection and modification are completed;
- int and mat are the two functions which, through different levels of data transformation, carry on the computation process: int = int1 ° int2 ° ... ° intn (° is a generic composition operator), and mat = mat1 ° mat2 ° ... matn.

This view 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.

4. Extending the concept of characteristic pattern

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 $cp_0 = <cs_0, u_0, <int_0, mat_0>>$. 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 pair of *virtual* css, of states of the process u, and of int and mat functions. This can be formalized by specifying, for each component in a $cp_i = <cs_i, u_i, <int_i, mat_i>>$ in the structure, a process of *translation*, *interpretation*, or *combination* from level i to level i-1.

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, $cs_i = g_i(cs_{i+1})$, 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_i of a higher level cp. Let us call f_i the interpretation function which allows the interpretation of a computational construct u_{i+1} at a higher level by a computational construct at a lower level u_i. Therefore, u_i is equal to $f_i(u_{i+1})$.

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. At level i, int_i is equal to $int_n \circ int_{n-1} \circ ... \circ int_{i+1}$ and, in a similar way, mat_i is equal to $mat_n \circ mat_{n-1} \circ ... \circ mat_{i+1}$, where n is the highest conceptual level.

At the lowest level, the virtual characteristic pattern cp_0 corresponds to the real characteristic pattern as defined above, that is to the state of an entity 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 g_0, g_1, ..., g_{n-1}, i.e.,

$$cs = cs_0 = g_0(g_1(...g_{n-1}(cs_n))),$$

where cs_n is the virtual cs considered at the highest conceptual level. Similarly, the state of the computational construct u_0 is obtained by a chain of interpretations through the application of functions f_0, f_1, ..., f_{n-1}, and the interpretation and materialization functions are obtained by the combination of all interpretation and materialization functions in the chain.

This generalization is justified also in terms of the Interaction Modelling Space. Each point of the Interaction Modeling Space IMS (i,j,k) identifies a VIM_{ijk}, i.e. a triple of functions (d_i, h_j, g_k) which specify the interpreters of programs, activities and css, each one at the respective level of abstraction, namely i, j or k level. In particular, on the program dimension, at level i, a program p_i is equal to $d_i(p_{i+1})$, at level i-1, a program p_{i-1} is equal to $d_{i-1}(p_i)$, and so on. In the similar way, on the activity dimension, at a level j, an activity a_j is equal to $h_j(a_{j-1})$, and on the cs dimension, at level k, a characteristic structure cs_k is equal to $g_k(cs_{k-1})$.

5. Conclusion

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.

6. References

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

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

[3] F. Pittarello, A. Celentano, "Interaction Locus: a Multimodal Approach for the Structuring of Virtual Spaces, *HCITALY 2001, Human-Computer Interaction Symposium*, Florence, Italy, 2001.

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

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