A Methodology for Selecting Visual Representations in Scientific and Simulation Applications

CARLA MARIA DAL SASSO FREITAS
FLAVIO RECH WAGNER

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Caixa Postal 15064 CEP 91501-970
Porto Alegre, RS, Brasil
carla@inf.ufrgs.br
flavio@inf.ufrgs.br

Abstract. This paper presents a methodology for selecting visual representations during the development of simulation studies or visualization applications. The methodology considers the classes of entities in a model, the classes of graphical presentation techniques, and the classes of queries that may be submitted by a final user about the model and/or data produced by simulation or analysis tools. The methodology is generic and may be applied to the visualization of data collected from many sources.

1 Introduction

Since the late 70's a growing number of systems has incorporated facilities to visually represent models and simulation results as well as to allow user interaction during the simulation. As to their degree of user interaction during the execution of simulation experiments, tools can offer visualization as post-processing, tracking or steering facilities (fig.1) [MAR 90]. Post-processing (fig. 1a) only allows the user to graphically visualize the results of a simulation execution after all the data had been calculated or collected. No interaction with the source of the data is possible. Depending on the data the visualization can be more or less interactive: in a sequence of scenes, the user might advance or go back in the pool of images; in visualizing a tridimensional scene, the user might alter the point of view in order to improve the observation of the phenomenon under simulation. By tracking, one understands the visualization of simulation results during the execution, but without interaction, except the possibility of aborting the simulation (fig. 1b), while steering is the direct control of the simulation during the execution, thus allowing the user to change parameters and immediately visualize the effects of that action (fig. 1c).

While the concept of steering was conceived in the framework of continuous systems, where the model is generally represented by a set of equations, visual interactive simulation [HUR 76] appeared in the context of discrete-event simulation, where the model is described as a set of inter-related entities. Both, however, are concerned with providing visual representation of data and interactive facilities during execution, which are important goals in scientific computing [McC 87] because scientists base their decisions on the exploratory analysis of the data, testing various hypothesis and alternatives.

Regarding visualization, the major problem is related to the mapping between data and images, i.e., the selection of adequate visual representations. Rarely the designer takes into account that the process of perception is subjective, complex, and involves several variables. One of the fundamental principles in the development of visual interactive simulations is "get the user involved as soon as possible" [BEL 87] coincident with the commitment with the user needs in user interface design [SHN 87]. This minimizes some future problems but there is an increasing need for a methodology to support the selection of visual representations, depending on the entities, the information needed about the entities, the operations to be accomplished with the data, different techniques of graphical presentations, etc.

This paper presents a methodology that considers the classes of entities in the model, the classes of graphical presentation techniques, and the classes of queries that may be submitted by a final user about the model and/or data derived from the model. The visualization techniques consider perception issues necessary to construct images that carry the relevant information in an effective way. The methodology is generic and may be applied to the visualization of data collected from many sources, i.e., raw data gathered by instruments, results of scientific data analysis or results of simulation studies.

2 Related work

There are few efforts addressing methodological aspects in simulation studies or even in more general visualization applications - see, for example, [ELS 88] and [ROB 91] - contrasting with the great number of reports on techniques for generating images of the chosen representations (see [BRO 92]).

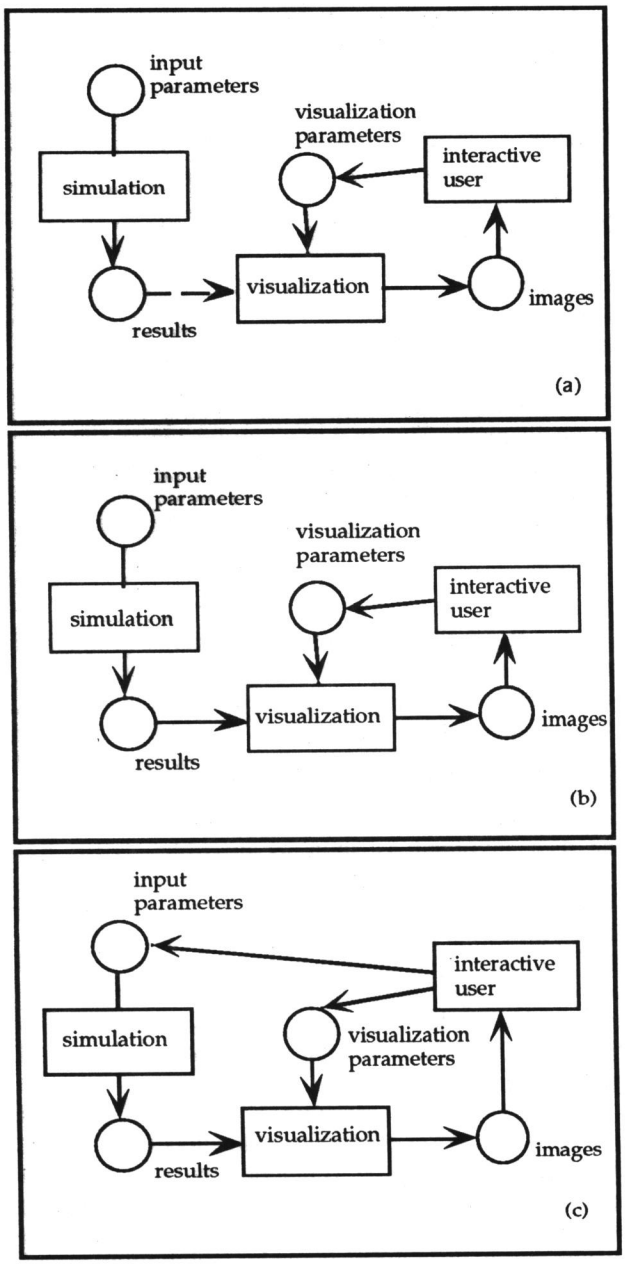

Figure 1- Types of visualization facilities [MAR 90]: (a) post-processing; (b) tracking; (c) steering.

The need of a methodology was identified early in the 80's, with the introduction of graphical facilities in existing discrete simulation languages and systems. Bell and O'Keefe [BEL 87] pointed out that visual interactive simulation needs a methodology because users may tend to evaluate simulation results looking only to the display, ignoring traditional statistics. Since the user may alter the model at his own, more risky would be the case of changing the model until it works as the ideal situation and not the real one. These authors pose four recommendations in the development of visual interactive simulation: 1) get the user involved as soon as possible; 2) build the visual model before the mathematical model; 3) design the interaction as general as possible; and 4) try to transfer the simulation to the end user, allowing model building based on pre-defined modules. The different phases of a simulation study [BAL 90] must be addressed in such a methodology because visual representations may be used for showing both the communication model and the (statistical) results of the experiments [JOH 88].

While in discrete systems simulation, entities may be easily identified and represented by icons in a way that even observers not familiarized with the problem can understand the situation, complex phenomena do not map immediately to a visual representation and require the use of carefully chosen abstractions. This selection may be done following guidelines as those described by Elson and Cox [ELS 88] for the visualization of injection molding in post-processing stage: 1) only present the relevant information, i.e., only the fundamental variables for the posed question; 2) use the shape of icons to represent vectorial quantities; 3) use color to represent scalar values; 4) use animation to represent evolution in time, and 5) use different rendering techniques to represent global threshold changes in the phenomenon. A different approach is the use of icons associated with cursors, specially in 3D space, allowing spatial interaction in quantitative queries about the data [KER 90]. Wehrend and Lewis [WER 90] do not establish principles for the construction of visual representations, but describe an approach based on classes of objects and classes of operations. By observing the data, classes of objects are identified (scalars, positions, regions, etc); by determining the user goals in visualizing the data, classes of operations are also identified (reading variables, comparing variables, etc). A matrix objects x operations results, and a given problem may be subdivided into sub-problems, populating the cells of the matrix. A catalog of visualization techniques (actually visual representations) suitable for the objects and operations is formed when one fills the matrix with techniques. The visualization expert may select a technique for a given pair (object, operation) consulting this catalog.

The natural scene paradigm was chosen by Robertson [ROB 90] as the basis of a methodology for the selection of visual representations in scientific visualization because human visual mechanisms are developed to understand the 3D real world, i.e., 3D structures and scenes. Even observing bidimensional representations, visual abilities guarantee the perception of a 3D scene [HAB 82]. Like Wehrend and Lewis [WER 90], Robertson proposes the analysis of data and user goals. However, his methodology can be better applied to multidimensional data. The types and dimensionality of data variables are identified and the interpretation needs of the users are reduced to a set of general characteristics which are described by means of

attributes of variables. In the natural scene paradigm, properties of the scene represent attributes of variables to be visualized. The identification of these properties is made in the same way as the data analysis, resulting a similar structure (data type, dimensionality, characteristics). Properties like surface height, surface covering material, material type, material condition, etc are used to describe physical attributes of a 3D scene. The mapping *data attribute - scene property* is direct, facilitating methodology automatization.

3 The bases of the methodology

As mentioned earlier a visualization methodology intended to be generic, i.e., to be applied to both simulation results and data gathered by different instruments, should consider the classes of entities and data involved, the classes of visual representations, and the classes of queries and interpretation needs. Actually, such a methodology should help the process of selecting an adequate visual representation for some pair {data, query}. The following sections present our understanding of such classes.

3.1 Characterization of entities and data

Usually simulation studies are based on inter-related entities owning properties (attributes represented by variables) or phenomena whose behavior may also be represented by properties varying during the simulated time interval. Often literature classifies data based on characteristics as data type, dimensionality, rank, etc [BRO 92], [ROB 90], [TRE 92].

Taking a generic approach, at first we classify the objects under study (see figure 2, at the end of the paper); after we address their static or dynamic attributes (properties). Objects in the model may be included in one of three classes, depending on the nature of the entities they represent, namely, discrete systems, manufacturable objects, real world entities or phenomena. Further, objects are characterized by attributes which in turn may be classified according to different criteria. In general, an object (described by data) is considered as a field or parameter of one or more values that is a function of one or more variables [BRO 92], [TRE 92]. We categorize attributes by the type, dimension, nature of their domain, and their distribution over the domain (figure 3). The paragraphs below explain how these categories may be interpreted.

The *data type* of an attribute indicates the classes of values that it can assume. An attribute is classified as a type *point* when the data corresponds to just a characteristic; there is no associated function. This is the case of the height of individuals of some population or the predominant flora of some region. *Point* attributes can assume numerical or nominal values. An attribute of type *scalar* denotes data that is sampled from a function over the domain of the attribute; examples are material density in medical data, temperature or pressure in a flow of some gas. A type *vector* attribute enables the representation of vectorial quantities as the velocity of winds or other flows. So, while a *scalar* is a single integer or real value, a *vector* is a n-uple of integer or real values. Attributes of type *tensor* arise in application areas such as computational fluid dynamics and finite element stress analysis; a second order tensor in 3D, for example, is represented by a matrix of nine components. A *parameter set* represents a collection of attributes that must be considered together in relation to another attributes. For example, the color of a pixel in an image is a parameter set (R,G,B). The type *aggregation* is used to classify attributes that are collections or organizations of data of any type. The three broad classes mentioned earlier (fig. 2) would rise to different interpretations of this type. For a model of a discrete system, this kind of attribute would contain the graphical description of the hierarchy or net of elements; this embodies the positions of icons representing elements and their connections. For a geometrically modeled object, the value of this sort of attribute would be some kind of geometric description of the object (polygons describing a mechanical part, for example). Finally, for a model of a real world entity this attribute would be used to define the mesh that specifies how values of other attributes map to some physical domain (for example, meteorological data about an area describing winds, pressure, temperature, and rain). An attribute of type *aggregation* may also be used to represent a series of data in time, i.e., the object may be represented as having a unique attribute of type *aggregation* constituted by n sets of data, each set corresponding to an observation.

The *dimension* of an attribute indicates if the value is defined over a 1D, 2D, 3D or higher dimensional space. Examples of 1D domain are distances measured from a point or some characteristic observed for an entity along a period of time. Classical 2D and 3D data are values observed for geographical (planar) and spatial areas, respectively. Examples of higher dimensional data may be taken from applications that generate multivariate data: remote sensing applications and statistical data about populations. The *dimension* of an attribute enables the representation of spatial, temporal, spectral and higher dimensional data.

The *nature of the domain* indicates if an attribute can assume values from an *enumerated set* or from a *continuous domain*. For continuous domains, the values may be *distributed* either *discretely* or *continuously*. Further a discrete distribution may be *pointwise* or over a *delimited region*. Considering a geographical area, an attribute that designates the height of the terrain is defined pointwise over a continuous domain, while another indicating the density of population in subareas is defined for regions rather than pointwise. Considering the fauna inhabiting this area, the number of individuals of each species is defined over an enumerated set; on the other hand, the total fauna density is defined continuously over all the region.

In the above discussion there is an important point that must be well understood, namely, the difference between domain dimension and cardinality of a parameter set. We refer to domain dimension as the number of independent variables of a function representing an attribute; if one of these variables is in reality a function, this variable is a parameter set and its cardinality is the domain dimension of the function. For example, in describing an image, we have F(x,y,color), where color is a parameter set (R,G,B). So, the dimension of the image is 2 and we are sampling an attribute *color* of type *parameter set*, with cardinality 3, over a 2D domain.

3.2 Classifying visual representations

Based on earlier studies [FRE 90], [FRE 92] we found close relationships between visual programming languages [CHA 87],[MYE 90] and visual simulation models, as well as, between program visualization and algorithm animation [BRO 88],[MYE 90] and visualization of simulation execution [MEL 85]. The last relationship was also pointed out by the participants of the *Workshop on Scientific Visualization Environments* [BUT 92].

In general, programs may be visually represented by several kinds of diagrams, forms (tables), and icons; for example, see [MOR 86], [SHU 85], and [GLI 90]. As to visual representations of simulation models, the first two classes are frequently used: block diagrams in GPSS/PC [COX 87] and in constructing SIMAN models [STU 90]; networks of icons in Slamsystem [LIL 90], Slam II [ORE 90], and Witness [GIL 89]. Forms are used to specify experiments in SIMAN simulations. As to program visualization and algorithm animation, including data structures presentation, the iconic and diagramatic approaches are usually employed; see [BRO 88] and [FOL 86] for examples of iconic representations; [KRA 83] and [REI 84], for diagramatically monitoring program execution. Simulation systems that incorporate presentation of simulation execution or visualization of simulation results in post-processing usually use the iconic approach, letting to the user the construction of graphic representations of objects and scenarios. This can be seen in Witness, SIMAN/CINEMA [POO 90], Simscript II.5 with Simgraphics [CAC 91], and Slam/Tess.

Both the visualization of models of complex phenomena and the visualization of scientific data in general do not follow the paradigms cited above. Dynamic systems may be modeled in some simulation languages and packages using diagrams, but the output of experiments are better interpreted if presented in another way. We have already cited the works of Robertson [ROB 90], formalizing the natural scene paradigm for visualizing scientific data using 2D and 3D structures, and Kerlick [KER 90] discussing the advantages of using iconic objects associated to 3D cursors specially for quantitative queries about the data. In a more general sense, Lohse et al. [LOH 90] classified visual representations depending on the type of knowledge they convey. Five principal groups of visual representations arose from a cluster analysis of sorting data: graphs and tables, maps, diagrams, networks, and icons. The visual representations were chosen from popular books on graphics and visualization, but they did not include some typical images of scientific data, as stress information mapped onto a 3D mechanical structure or the trajectories of flow around an aerofoil.

Our classification considers that a visual representation intends *to represent an entity or a phenomenon* or *to represent informations about them* (structure, static properties, behavior or any combined informations). Based on addressing these requirements, and observing visual representations employed in several applications, the classification summarized in figure 4 arises.

These classes of visual representations embodies several ones provided by visualization techniques as those described by Brodlie et al. [BRO 92]. Some examples are a bar chart (*graph*) representing the values of some attribute over time intervals; a 3D graph (class *graph*) showing in its height the value of some attribute of a 2D region; an *image* representing with different colors, different values of an attribute in some space; a *diagram* of *icons* representing the different entities in the structure of a system. The representation of the behavior of an object may be made by taking a sequence of changing visual representations (any of the above ones), and exhibiting it either in real time or in playback.

Regarding visual representations, another point deserves our attention. Visual programming languages are not only being used to represent simulation models: visual programming paradigms [AMB 89] are the current trend in the construction of user interfaces in modern visualization environments. The dataflow paradigm used in AVS [UPS 89], apE [DYE 90], and IRIS Explorer [SIL 91] is an example.

3.3 Addressing user's goals

The selection of a visual representation depends on the class of information the user needs to obtain from the data. While Robertson [ROB 90],[ROB 91] bases his methodology on three distinct interpretation aims depending on the user's needs - knowing the values at a point, the local distribution of values, and the global distribution of values -, Wehrend and Lewis [WER 90] distinguish visualization problems as classes of operations, namely, identify, locate, distinguish, categorize, cluster, distribution, rank, compare, relate, associate, and correlate.

We distinguish the following user's goals in visualizing objects, i.e., entities, systems, or data in general:

1. identifying an object in a context;
2. analyzing the structure of objects;
3. analyzing static properties of objects;
4. analyzing dynamic properties of objects;
5. tracking the behavior of objects.

The identification of an object in a context is, for example, the goal of a user studying the behavior of individual entities in systems like supermarkets, banks, etc. In another class of applications, as surgical planning, identifying an object may be the depiction of the exact region to be extirpated by the surgeon.The analysis of the structure of a complex object is an usual goal in applications like simulation or analysis of network systems, modeling mechanical systems, or studying organs reconstructed from medical data.Distinguishing static properties is one of the major goals of scientific users: data gathered by instruments or produced by simulation or analysis tools are explored by the user searching for patterns or substructures (regions with same temperature values, for example). Analogously, the analysis of dynamic properties of the objects are another major concern of scientists or engineers: evolution of dynamic properties in time like the velocity of wind in a certain region or the length of queues in a bank simulation are examples from distinct application areas.The fifth user goal in visualization, namely tracking the behavior of an object, requires the display of images that dynamically depicts the changes occurring in the object. Some examples are a coal mine or a service bay in a railroad, and the reproduction of a meteorological phenomenon.

Some of these user's goals may be further refined in more specific needs. For example, analyzing the structure of a complex object may involve to locate elements and substructures or to categorize/cluster them. Distinguish/analyzing static or dynamic properties include operations like to compare, relate, correlate, and associate values of variables. Also, questions may be posed by a user about numerical values, without the need of conveying them to visual forms.

4 The methodology

The proposed methodology is not based on a paradigm chosen as the basic visual representation. In order to be generic it is based on the three subjects already discussed. We build a double entry table {class of data, class of query}, which maps to adequate visual representations (fig. 5). The first three steps in the methodology correspond to the classical phases of determining the objects in the user's universe and the operations he/she needs to perform on the data: acquire knowledge about the data, determine user's goals; and refine user's goals in terms of queries. After these steps, specific phases leads to the selection of visual representations: match the {data, query} pair with one (or more than one) of the visual representations; and establish the valid composition of queries according to the chosen visual representations.

1. Acquire knowledge about the data:

The classes of objects (according to figure 2) must be identified as well as their attributes in terms of type, dimension, nature, and distribution (figure 3).

2. Determine user's goals in terms of the necessary queries:

a) questions concerning identification, i.e., which objects need individual identification;
b) questions concerning analysis of structure: systems will be structurally described in terms of entities and relationships, physical objects will be defined by a geometric model, and collections of data must be distributed over regular and irregular grids;
c) questions concerning static properties of objects and data, represented by variables: what are the static properties, how are they distributed over the object, what are the type, dimension, and nature of variables;
d) questions concerning dynamic properties of objects, systems, and data: what are the dynamic properties, how are they distributed, what are the type, dimension, and nature of variables;
e) questions concerning tracking/steering the behavior of objects and data: what variables may be queried, queries with visual results, queries with numerical results.

3. Refine user's goals, listing for each query:

a) graphical and/or numerical results;
b) objects or variables to which the query applies;
c) parameters for processing the query;
d) other queries with which the query in hand can be combined.

4. Match the [data, query)]pair with one (or more than one) of the visual representations (fig. 5)

a) identification of the object or entity in a context: icon;
b) representation of the structure of the object:
   - block diagram (complex object)
   - icon diagram (multiple-entity system)
   - geometric model (physical object)
   - realistic model (physical object)
c) representation of static properties of objects
   - scatter plot, to depict objects when they are described by multivariate data;
   - line chart, bar chart, and histogram, to depict values of properties of objects;
   - maps or images, representing (on the geometric model):
     . isovalues as line-based or discrete-shaded contouring;
     . surface-based contouring of regions of isovalues;

- 3D graph depicting values of variables either in wireframe or as a shaded surface;
- 2D/3D graph with icons: each icon may incorporate several variables (color, height, shape, texture, type of rendering)

d) representation of dynamic properties of the objects:
- the same visual representations cited above may be used for representing dynamic properties considering them as functions of time;
- they also may be used to take snapshots of the state of the system at specific instants of time;
- the trends in evolution of values of variables may be represented by features of icons in maps or images.

e) tracking the behavior of objects and systems is obtained through controlled animation of visual representations.

5. Establishing the valid composition of queries as to the chosen visual representations

This can be understood as composition constraints, because they determine what visual compositions are perceptually unambiguous.

The methodology may be employed by a user of a visualization system, who needs to choose visual representations for some data, and also, by a system designer developing a specific visualization application. The latter will employ the methodology to choose visual representations in order to implement the corresponding visualization techniques. Two important situations must also be mentioned. First, instead of choosing a single visual representation for a pair {data, query}, the system designer may select more than one representation in order to offer alternatives in an interactive interface. In this case it's up to the final user of the system to select the visual representation for a query. The second situation arises when the volume of data is extremely large: it will be necessary to have hierarchical visual representations in such a form that the final user can visually explore the data, selecting subsets (data culling [SPR 92a]), and zooming in and out. Since these hierarchical representations are also based on attributes of objects and user's goals, the methodology can also be employed to select them.

5 Final remarks

The work focuses the development of a methodology for choosing visual representations. The two first steps are classical phases in software engineering since they are concerned with requirements analysis. Step 3 is found in methodologies for interface design. Steps 4 and 5 are the specialized phases for choosing visual representations.

The methodology was successfully employed in the development of EFVis, a visualization system for the study of electromagnetic fields involving given objects [SCH 93]. EFVis has an interactive interface which offers facilities for selecting object and field attributes to be observed; the geometric model of an object may be displayed with visual representations of attributes depicted on it.

We are presently working on the development of an object-oriented framework associated with a visual language that supports this methodology by enabling the selection of data and object representations both for simulation studies and visualization of multivariate functions and data [FRE 93]. This framework is being conceived considering all the activities involved in simulation studies and data analysis process, i.e., it must support operations other than mapping data to images. Excellent discussions about phases of the simulation process and activities in scientific data analysis can be found in [BAL 90] and [SPR 92b], respectively.

6 Acknowledgments

We gratefully acknowledge Ana Elisa F. Schmidt and Adroaldo Raizer for employing a first draft of the methodology in developing EFVis. This work has been supported by CNPq and FAPERGS grants.

7 References

[AMB 89] Ambler, A.L. & Burnett, M.M. Influence of the visual technology on the evolution of language environments. *Computer, 22*(10):9-22, october 1989.

[BAL 90] Balci, O. Guidelines for successful simulation studies. In: Winter Simulation Conference, 1990. pp. 25-32.

[BEL 87] Bell, P.C. & O'Keefe, R.M. Visual interactive simulation - history, recent developments, and major issues. *Simulation, 49*(3):109-116.

[BRO 92] Brodlie, K.W. et al. *Scientific visualization - techniques and applications.* Berlin, Springer-Verlag, 1992.

[BRO 88] Brown, M.H. Exploring algorithms using Balsa-II. *Computer, 21*(5):14-33, may 1988.

[BUT 92] Butler, D.M. & Hansen, C. Visualization'91 workshop report: scientific visualization environments. *Computer Graphics, 26*(3):213-216, august 1992.

[CAC 91] CACI Products Co. *SIMGRAPHICS User's guide and casebook.* La Jolla, CA, 1991.

[CHA 87] Chang, S.K. Visual languages: a tutorial and survey. *IEEE Software, 4*(1):29-39, january 1987.

[COX 87] Cox, S. Interactive graphics in GPSS/PC. *Simulation, 49*(3):117-122, september 1987.

[DYE 90] Dyer, D.S. A dataflow toolkit for visualization. *IEEE Computer Graphics & Appl., 10*(4):60-69, july 1990.

[ELS 88] Elson, I.S. & Cox, D. Visualization of injection molding. *Simulation, 51*(5):184-188, november 1988.

[FOL 86] Foley, J.D. & McMath, C.F. Dynamic process visualization. *IEEE Computer Graphics & Appl., 6*(2):16-25, march 1986.

[FRE 90]Freitas, C.M.D.S. *Visualization techniques for simulation applications*. Porto Alegre, UFRGS, july 1990. (technical report TI 187)

[FRE 92] Freitas, C.M.D.S. *A systematic study on visual languages*. Porto Alegre, UFRGS, september 1992. (technical report RP 197)

[FRE 93] Freitas, C.M.D.S. *Metodologia e Técnicas para Visualização de Dados Multivariados*. Porto Alegre, PGCC/UFRGS. Tese de Doutorado, em and.

[GIL 89] Gilman, R.A. & Billingham, C. A tutorial on SEE WHY and WITNESS. Winter Simulation Conference, IEEE, 1989. pp. 192-200.

[GLI 90] Glinert, E.P.; Kopache, M.E. & McIntyre, D.W. Exploring the general-purpose visual alternative. *Journal of Visual Languages and Computing, 1*:3-39, 1990.

[HAB 82] Haber, R.N. & Wilkinson, L. Perceptual components of computer displays. *IEEE Computer Graphics and Appl., 2*(5):23-35, 1982.

[HUR 76] Hurrion, R.D. *The design, use and required facilities of an interactive visual computer simulation language to explore production planning problem*. PhD thesis. University of London, England, 1976.

[JOH 88] Johnson, M.E. & Poorte, J.P. A hierarchical approach to computer animation in simulation modeling. *Simulation, 50*(1):30-36, january 1988.

[KER 90] Kerlick, G.D. Moving iconic objects in scientific visualization. In: IEEE Conference on Visualization, 1990. pp. 124-130.

[KRA 83] Kramlich, D.; Brown, G.P.; Carling, R.T. & Herot, C.F. Program visualization graphics support for software development. In: Design Automation Conference, 1983. pp. 143-149.

[LIL 90] Lilegdon, W.R. & Ehrlich, J.N. Introduction to SLAMSYSTEM. In: Winter Simulation Conference, 1990. pp. 77-79.

[LOH 90] Lohse, J.; Rueter, H.; Biolsi, K. & Walker, N. Classifying visual knowledge representations: a foundation for visualization research. In: IEEE Conference on Visualization, 1990. pp. 131-138.

[MAR 90] Marshall, R.; Kempf, J.; Dyer, S. & Yen, C. Visualization methods and simulation steering for a 3D turbulence model of lake Erie. *Computer, 24*(2):89-97, march 1990.

[MCC 87] McCormick, B.H. DeFanti, T.A. & Brown, M. (eds.) Visualization in Scientific Computing. *Computer Graphics*, vol. 21, no. 6, november 1987.

[MEL 85] Melamed, B. & Morris, R.J.T. Visual simulation: the Performance Analysis workstation. *Computer, 18*(8):87-94, august 1985.

[MOR 86] Moriconi, M. & Hare, D.F. The PegaSys system: pictures as formal documention of large programs. *ACM Trans. on Programming Languages and Systems, 8*(4):524-546, october 1986.

[MYE 90] Myers, B.A. Taxonomies of visual programming and program visualization. *Journal of visual languages and computing, 1*:97-123, 1990.

[ORE 90] O'Reilly, J.J. & Whitford, J.P. SLAM II Tutorial. In: Winter Simulation Conference, 1990. pp. 73-76.

[POO 90] Poorte, J.P. & Davis, D.A. Computer animation with CINEMA. In: Winter Simulation Conference, 1990. pp. 123-127.

[REI 84]Reiss, S.P. Graphical program development with Pecan program development systems. *Sigplan Notices, 19*(5):30-41, may 1984.

[ROB 90] Robertson, P.K. A methodology for scientific data visualisation: choosing representations based on a natural scene paradigm. In: IEEE Workshop on Visualization, 1990. pp. 114-123.

[ROB 91]Robertson, P.K. A methodology for choosing data representations. *IEEE Computer Graphics & Appl., 11* (3):56-67, may 1991.

[SCH 93] Schmidt, A.E.F., Freitas, C.M.D.S. & Raizer, A. Interactive visualization of finite element method results in studying electromagnetic fields. In: COMPUMAG Conference, Miami, Florida, October 1993.

[SHN 87] Shneiderman, B. *Designing the user interface - strategies for effective human-computer interaction*. Reading, Mass., Addison-Wesley, 1987.

[SHU 85] Shu, N.C. FORMAL: a forms-oriented visual directed application development system. *Computer, 18*(8):38-50, august 1985.

[SIL 91] Silicon Graphics Inc. *Iris Explorer User's Guide*. 1991.

[SPR 92a] Springmeyer, R.R. *Designing for Scientific Data Analysis: from Practice to Prototype*. Livermore, CA, University of California, 1992. (Ph.D thesis)

[SPR 92b] Springmeyer, R.R., Blattner, M.M. & Max, N.L. A Characterization of the Scientific Data Analysis Process. In: IEEE Workshop on Visualization, 1992. pp. 235- 242.

[STU 90] Sturrock, D.T. & Pegden, C.D. Introduction to SIMAN. In: Winter Simulation Conference, 1990. pp. 109-114.

[TRE 92] Treinish, L.A. Introduction to data management methods for scientific visualization. In: *Introduction to scientific visualization tools and techniques*. SIGGRAPH'92 Course 1 Notes, 1992. pp. 6.1-6.26.

[UPS 89] Upson, C; Faulhaber, T.; Kamins, D.; Laidlaw, D.; Schlegel, D.; Vroom, J.; Gurwitz, R. & Van Dam, A. The Application Visualization System: a computational environment for scientific

visualization. *IEEE Computer Graphics & Appl.*, 9(4):30-42, july 1989.

[WER 90] Wehrend, S. & Lewis, C. A problem-oriented classification of visualization techniques. In: IEEE Workshop on Visualization, 1990. pp. 139-143.

Classes of objects	Definition	Examples
models of discrete systems	organized set of inter-related entities	computer network manufacturing system
geometrically defined models	existing or manufacturable object	mechanical part beam actor of an animated seq.
models of real world entities	structured or unstructured set of data an entity or a phenomenon	metereological data medical data populational data

Figure 2 - Classes of objects, interpretation and examples.

Characteristics of attributes	Meaning
data type	point (numerical or nominal) scalar vector tensor parameter set aggregation
dimension	1D 2D 3D nD
nature of the domain	enumerated set continuous domain
distribution over the domain	member (enumerated set) continuous discrete (pointwise or by regions)

Figure 3 - Characterization of attributes.

Visual representation	**Information conveyed**
icon	identification of the object in a context
graphs and tables	static properties of objects evolution of properties in time
maps and diagrams/networks	structure of complex objects static properties of objects snapshot of the behavior of objects
images	geometric model of an object static properties of objects static properties mapped onto the geometric model snapshot of the behavior of an object
sequence of maps, diagrams, networks or images	behavior of an object

Figure 4 - Classification of visual representations

	User goal or query				
Object	identification of the object	representation of structure	static properties	dynamic properties	accompanying behavior
entity in a system	*icon*	*icon*	*icon characteristics, graph*	*icon charact., graph*	*animation of chosen repres.*
system		*block diagram, icon diagram, network*		*graph*	*animation of chosen repres.*
physical object	*icon*	*geometric model, realistic model*	*graph, map*	*graph, map*	*animation of chosen repres.*
data taken from the real world	*graph, icon*	*map*	*graph, map*	*graph, map*	*animation of chosen repres.*

Figure 5 - Suggested visual representations for a given data and query pair.