6.3.3 Dynamic exploration environments

Robert Belleman[1], *Peter Sloot*[2]

Introduction: exploration environments

In many scientific computing problems, the complexity of both the simulation and the generated data is too vast to analyze analytically or numerically. For these situations, exploration environments provide essential methods to present and explore the data in a way that allows a researcher to comprehend the information it contains. Exploration environments combine presentation and interaction functions into one system to allow exploration of large data spaces. These data spaces may originate from data acquisition devices or represent results from computer simulations. In our research we discriminate between static and dynamic exploration environments.

In Static Exploration Environments (SEE), the data presented to the user is time invariant; once the data is loaded into the environment, the user is presented with a visual representation of this data. Interaction methods are provided to change the visualization parameters interactively in order to get the best view to gain understanding. The data itself, however, does not change (see Figure 1).

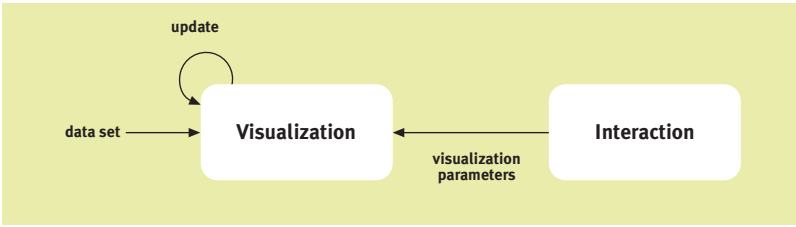


Figure 1
Schematic representation of a static exploration environment (SEE).

An important step towards a successful exploration environment is to involve the researcher in the presentation as much as possible, thereby increasing the researcher's level of awareness [Bryson, 1996a]. To achieve this, an exploration system needs the following, often conflicting capabilities:

- *High quality presentation*. The most common method to provide insight in large multidimensional data sets is to represent data as visual constructs that present quantitative and relational aspects to the observer in an comprehensible manner. Many scientific visualization environments are now available that provide means of efficiently achieving this [IBM, 1991; IrisExplorer, 1998; Schroeder, 1997; Upson, 1989].
- *High frame rate*. While the capabilities of modern graphical hardware allow increasingly complex images to be rendered with relative ease, the level of detail in the presentation should be minimized to avoid information clutter and achieve high frame rates (a compromise is often necessary). For an inter-

1 R.G. Belleman MSc,
robbel@wins.uva.nl,
The Universiteit van Amsterdam,
Faculty of Science, Section
Computational Science, Amsterdam,
The Netherlands

2 Prof Dr P.M.A. Sloot,
sloot@science.uva.nl,
The Universiteit van Amsterdam,
Faculty of Science, Section
Computational Science, Amsterdam,
The Netherlands

active exploration environment the visual frame rate should be at least 10 frames per second [Bryson, 1996b].

– *Intuitive interaction*. A prerequisite of a successful SEE is that a sufficiently rich set of interaction methods is provided that allows a user to extract both qualitative and quantitative knowledge from the data sets. An unfortunate side effect of increasingly richer sets of interactive methods is that user-friendliness is compromised, so careful consideration is required during user-interface design.
– *Real-time response*. Some delay will always occur between the moment a user interacts with a presentation and the moment that the results are visible. This is caused by low tracking rates of input devices, (re-computations, communication delays or temporary reduced availability of computational or network resources. To attain accurate control over the environment and to avoid confusing the user, the amount of lag in an exploration system should be minimized [Taylor, 1996].

Provided these capabilities are carefully considered, such environments are well suited for the exploration of static multidimensional data sets [Belleman, 1998]. Static exploration environments can be customized to observe iteratively updated data sets produced by 'living' simulations. When interaction with the simulation is also allowed, however, we speak of dynamic exploration environments and radically different considerations come into play, as we describe in the next section.

Dynamic exploration environments

Dynamic Exploration Environments (DEE) extend the previously described static model in such a way that the information provided to the user is regenerated periodically by an external process, in our case a computer simulation. Here, the environment is expected to provide (1) a reliable and consistent representation of the results of the simulation at that moment and (2) mechanisms enabling the user to change parameters of the external process (i.e. simulation) (see Figure 2).

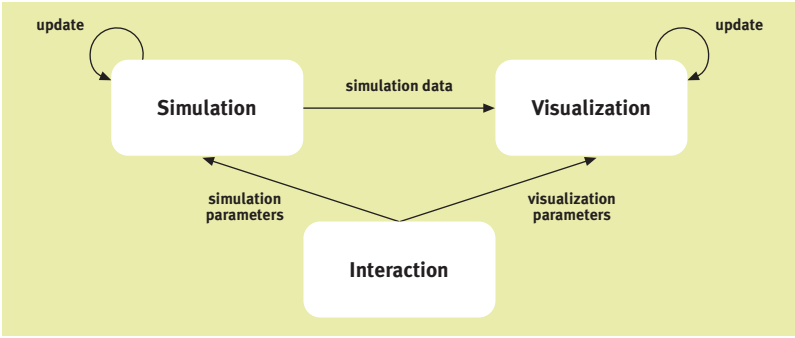


Figure 2
Schematic representation of a dynamic exploration environment (DEE).

Dynamic environments have additional requirements over static environments. For example, in static interactive systems, the interaction functions can be implemented inside the visualization environment, since the only interaction that takes place is with the visualization. In dynamic environments, interaction influences both the visualization and the simulation environment. Changing a static environment into a dynamic environment therefore requires that at least one module performs additional processing to service the interaction. Such a change makes these environment less suitable for use in other applications without significant modifications. In the sequel we address some of the functionalities required to develop generic dynamic exploration environments. We start with a brief description of the specific time management aspects in DEE and present a top-down description of the associated additional requirements in such systems.

Time management

An important issue in a DEE is time management. Time management deals with the exchange of time stamped information between components. For a DEE, the four most time demanding components are; the simulation environment, the visualization modules, the rendering layer and the explorer (i.e. the user).

Figure 3 and 4 show time frame diagrams illustrating the advancement of time, under two different time management strategies; lock-step and asynchronous. Time frames are illustrated by rounded boxes. The gaps between time frames on a same level represent the idle time of the component on that level. The gaps between time frames on neighboring levels represent the delays that occur between the time one component is done with a time frame and the next component starts working on it. These delays are delineated at the bottom of the Figure.

Figure 3
Time frames and delays in a lock-step interactive dynamic exploration environment.

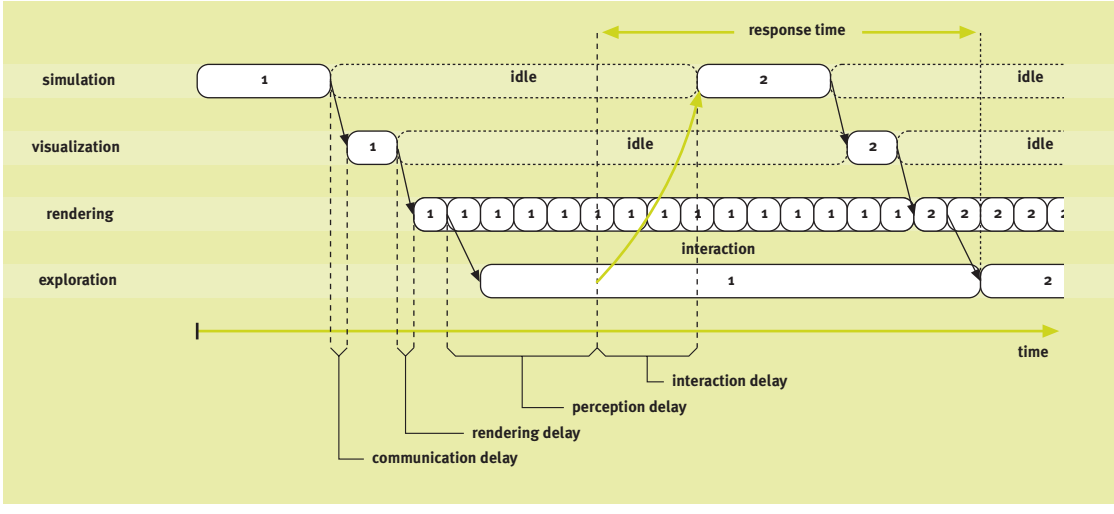


Figure 3 shows a time frame diagram in a lock-step interactive dynamic exploration environment. In this strategy, the simulation is allowed to advance, only if the user explicitly tells the environment it is alright to do so. While the user is exploring the results rendered by the graphics system, the simulation and visualization modules sit idle. In situations where a single simulation, visualization and rendering time frame takes a negligible amount of time, this strategy may be perfectly adequate, since the user will see the result of his interaction in short notice. However, if the these time frames are long, it may take a long time before the result of an interaction is shown. This often leads to an unusable environment, confusing the user.

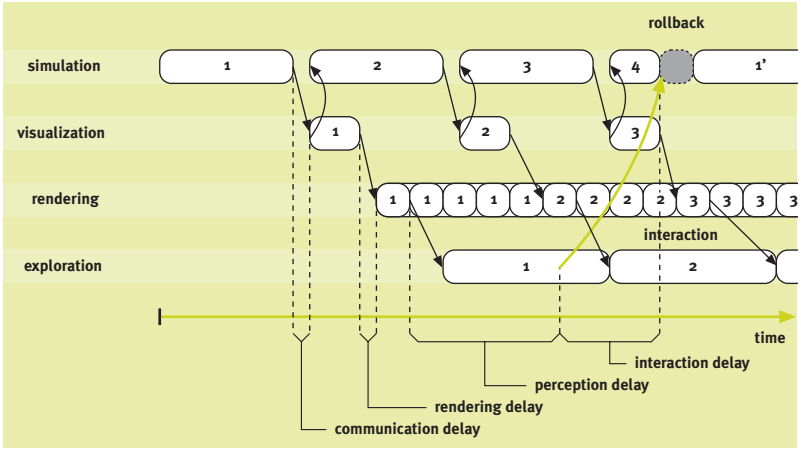


Figure 4
Time frames and delays in an asynchronous interactive dynamic exploration environment.

The time required before simulation updates are presented to the user (i.e. the length of a time frame on the exploration level) can be shortened by allowing the simulation to run asynchronously from the rest of the environment. Figure 4 shows a time frame diagram in an asynchronous environment. In an asynchronous system, different components are allowed to advance, when they have finished processing and communicating the current time frame. Different components may therefore execute at different time scales. In addition, these time scales are mostly non-deterministic because of hardware, software and human imposed delays. As a consequence, time delays occur as the output generated by one component cannot be accepted immediately by another component for processing. When components depend on the output of an increasing number of other components, the time frame that is processed by 'later' components fall further apart. This has a causality consequence for the user who explores the final component in the environment and therefore interacts with components at a much 'earlier' time compared to what is being processed by a simulation at the same wall-clock time. Time management is responsible for detecting and resolving this causality violation. Methods for resolving time causality problems have been investigated by [Overeinder, 1999].

Interaction

A DEE provides the opportunity to interact with a living simulation. This interaction can take any form; from typed input for simple types of interaction via graphical user interfaces to fully immersive virtual environments. The main feature of immersive environments over other graphical user interfaces is that user-centered stereoscopic images are presented to a user rather than visualization centered three-dimensional (3D) projections. User-centered stereoscopic images differ from projections on a flat screen in that slightly different pictures are generated for the left and right eye, dependent on the position of the viewer. This makes images 'pop out of the screen' and react to the user's movements[3], an important depth-cue to gain understanding in complex multidimensional structures.

A minimal requirement for interaction in an immersive Virtual Environment (VE, see also [Kaandorp, 1998]) is the availability of input devices that can be used to convey intention to the environment. The most common are sensor devices that measure the 6 Degrees Of Freedom (DOF) one has to move around in a 3D space (position and orientation). These sensors can be used to detect the proximity of a physical object (such as the user's hand) to virtual objects, so that the user can interact with them. Interaction with a virtual environment is a key issue, especially in an interactive simulation environment. The following types of interaction are deemed mandatory:

- *Object interaction*. An 'object' is defined here as a visual entity that is in the center of interest to an end-user. These objects are representations of data sets or simulation results, but can also be 'widgets' (menus, buttons, sliders, etc.). An object has attributes associated with it such as position, scale, level, state, etc. Object interaction is concerned with changing these attributes.
- *Navigation and way finding*. Navigation provides users with methods to move beyond the confinements of the VE's physical dimensions. Objects beyond the VE come into reach by moving the user towards them. Note how this concept places the user of the VE in the center of this type of interaction; the user is transported from one place to another, while the objects remain where they are. Way finding is a relatively new concept in VE applications and provides the user with a reference on where he is in a virtual environment [Elvins, 1997].
- *Probing*. Although visual presentations allow researchers to qualitatively analyze their data, an instrument for obtaining quantitative information from the visualization is a valuable asset. An architecture that allows researchers to probe visual presentations in order to obtain quantitative information is described in [Belleman, 1999].

3 This is commonly referred to as 'motion parallax'.

Coordination: intelligent agents

Since the various components in a DEE are independent processes, some means of coordination between them is required to allow a complex environment of this kind to be used. Especially in the case of interactive simulations, interaction involves not only the visualization element, but also the simulation part. For software engineering and efficiency reasons, it is reasonable to move the processing for those general interactions into independent modules, and let them be reusable to all components that need these interactions. One approach to this is through Intelligent Agents (IAs, see also Figure 5).

As are software modules with the capability of performing three functions: (1) perceiving state changes in the environment through the use of monitors, (2) taking actions that affect conditions in the environment and (3) reasoning to interpret perceptions, solve problems, draw inference, and determine actions [Hayes, 1995]. Agents execute autonomously, interfering minimally with the rest of the environment, apart from communicating with other agents or the user.

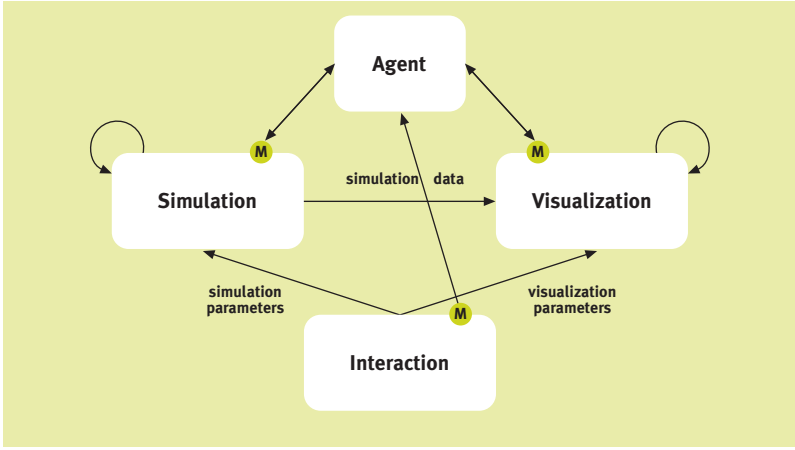


Figure 5
A DEE instrumented with monitors and an intelligent agent.

Feedback is generated when the agent has solved a problem, or has prepared suggestions based on the current running context. This feedback could be information to the user, or the information is sent to environment components. Depending on the permission settled beforehand, an agent could just provide feedback without affecting the working status of the environment, or make some changes in the environment based on its reasoning.
At present, agents are used to provide feedback to the user concerning the state of a simulation (e.g. accuracy of the simulation, time to completion, convergence rate) and user interaction (including speech recognition and synthesis). In the near future agents will be developed for feature extraction (e.g. determining the geometric skeleton of objects, detecting eddies in flow domains) and providing assistance (context sensitive help).

Distributed execution environment

Although the capabilities of modern computer systems are nearing the requirements for performing both simulation and visualization tasks on the same machine, some performance increase may be attained by running these tasks on dedicated computing platforms. For example, many simulation applications perform better on dedicated hardware such as vector processors, massively parallel platforms or other High Performance Computing (HPC) machinery, while state of the art graphical systems are now available that are well suited for the visualization tasks. Moreover, a decomposition of the environment into separate communicating tasks facilitates implementation and allows more control over the performance of the system as a whole (in Figure 5 each block can be considered a separate process or a combination of processes, possibly running on different systems).

Especially in the case of distributed environments, some means of job control is required that starts/stops the execution of the different components of the environment on the various computing platforms. In many organizations this system also needs to allocate the required resources prior to execution (for example in the case of batch execution systems). GLOBUS is one such software infrastructure for computations that integrates geographically distributed computational and information resources [Foster, 1997].

In distributed systems, components execute on different, possibly heterogeneous computing platforms. To be able to communicate data with each other, components provide access to their attributes, which can then be made available to other components. In heterogeneous computing environments the attributes often have to be converted into different representation formats. Furthermore, in many circumstances not all components in an environment will participate in a communication. For these situations a publication and subscription mechanism needs to be provided that limits communication to members of a restricted group.

Attribute ownership

The behavior of individual components in the environment is defined by one or more attributes (or parameters), which together define the state of that component. In a distributed system attribute changes (which can be considered to be events, for example as a result of user interaction) should only be performed by a component that owns the attribute to avoid race conditions. In some cases it may be necessary to transfer ownership so that attributes can be changed by other components (for example in a collaborative environment where multiple users manipulate the same components).

Runtime support system

From the considerations described in the previous sections, it becomes clear that a generic framework to support the different modalities is required. In our research we have chosen for the 'High Level Architecture' (HLA) as a suitable architecture for constructing a DEE.

HLA provides solutions to many of the problems and issues described in the previous sections. Specifically, HLA allows data distribution across heterogeneous computing platforms (including message groups), supports a flexible attribute publish/subscribe and ownership mechanism and offers several methods to do time management.

Vascular reconstruction: a case study

The design considerations described in the previous section cover the issues that are involved with building a DEE. The architecture is validated by analysis of a prototypical case study of simulated abdominal vascular reconstruction.

The application we have chosen as a test case combines visualization, simulation, interaction and real-time constraints in an exemplary fashion. By a detailed analysis of the spatial and temporal characteristics of the test case we attempt to recognize generic elements for the design of a computational steering architecture. We begin with a description of the test case.

Simulated abdominal vascular reconstruction

Vascular disorders in general fall into two categories: stenosis, a constriction or narrowing of the artery by the build-up over time of fat, cholesterol and other substances in the vascular wall, and aneurysms, a ballooning-out of the wall of an artery, vein or the heart due to weakening of the wall. Aneurysms are often caused or aggravated by high blood pressure. They are not always life-threatening, but serious consequences can result, if one bursts.

A vascular disorder can be detected by several imaging techniques such as X-ray angiography, MRI (Magnetic Resonance Imaging) or Computed Tomography (CT). Magnetic Resonance Angiography (MRA) has excited the interest of many physicians working in cardiovascular disease, because of its ability to non-invasively visualize vascular disease. Its potential to replace conventional X-ray angiography methods which use iodinated contrast has been recognized for many years, and this interest has been stimulated by the current emphasis on cost containment, outpatient evaluation, and minimally invasive diagnosis and therapy [Yucel, 1999].

A surgeon may decide on different treatments in different circumstances and on different occasions, but all these treatments aim to improve the blood flow of the affected area . Common options include thrombolysis where a blood clot dissolving drug is injected into, or adjacent to, the affected area using a catheter; balloon angioplasty and stent placement, which is used to widen a narrowed vessel by means of a inflatable balloon or supporting framework; or vascular surgery. A surgeon resorts to vascular surgery, when less invasive treatments are unavailable. In endarterectomy the surgeon opens the artery to remove plaque build-up in the affected areas. In vascular bypass operations, the diseased artery is shunted using a graft or a healthy vein harvested from the arm or leg.

The purpose of vascular reconstruction is to redirect and augment blood flow, or perhaps repair a weakened or a neurysmal vessel through a surgical procedure. The optimal procedure is often obvious, but this is not always the case, for example, in a patient with complicated or multi-level disease. Pre-operative surgical planning will allow evaluation of different procedures a priori under various physiologic states such as rest and exercise, thereby increasing the chances of a positive outcome for the patient [Taylor, 1998].

What is needed?

The test case described in the previous section contains all aspects of an interactive dynamic exploration environment that are of consequence in the construction of a generic dynamical computational steering architecture. Our aim is to provide a surgeon with an environment in which he or she can try out a number of different bypass operations and see the influence of these bypasses. The environment needs the following:

- An environment that shows the patient under investigation with his infliction. A patient's medical scan is 3D, so to obtain best understanding on the nature of the problem the surgeon should be able to look at his specific patient data in 3D, using unambiguous visualization methods.
- An environment that allows the surgeon to plan a surgical procedure. Again, this environment should allow interaction in a 3D world, with 6 DOF. The CAVE environment allows us to interact with 3D computer generated images using 6 DOF interaction devices [SARA, 1998; Cruz-Neira, 1993]. Note that visual realism is not the primary goal here; what is more important here is physical realism, and then only of particular issues in fluid flow, as discussed later[4].
- An environment that shows the surgeon the effect of his planned surgical procedure. As the aim of the procedure is to improve the blood flow to the affected area, the surgeon must have some means to compare the flow of blood before and after the planned procedure. This requires the following:

4 This in contrast to research projects towards virtual operating theatres that include the simulation of tissue deformation and realistic blood spills [Basdogan, 1999; Bockholt, 1999].

- a simulation environment that calculates pressure, velocity and shear stress of blood flowing through the artery;
- a visualization environment that presents the results of the simulation in a clear unambiguous manner;
- an exploration environment that allows the researcher to inspect and probe (qualitatively and quantitatively) the results of the simulation (e.g. means for performing measurements, annotate observations, releasing tracer particles in the blood stream, etc.).

All this should be interactive, or in other words, it should be fast enough in such that a surgeon does not have to wait for the simulation results.

Implementation of a dynamic exploration environment

Parts of the components mentioned in the previous section have already been implemented in the course of previous projects. Others require minor adaptations to fit into our dynamic exploration architecture. In the following subsections we will briefly discuss the current status of the visualization and exploration environment, the interaction environment, the simulation environment and the middleware that combines these together.

VRE: immersive static exploration

We have previously built a SEE called the Virtual Radiology Explorer (VRE [Durnford, 1999; Versweyveld, 1998]) which is capable of visualizing medical CT and or MRI data in 3D (see also Figure 6). 3D data sets acquired with CT or MRI are often displayed and evaluated from various perspectives or at different levels, including sets of single slices, stack mode (cine loop) interactive representation of sets of slices, or Multi-Planar Reformation (MPR) represented as single slices or interactive cine loops. Despite the increased possibilities of acquiring data, clinical use of 3D rendering has been hampered by insufficient computing capacity in the clinical environment.

An example of the clinical use of 3D rendering is simulated endoscopy[5]. Simulated endoscopy has several advantages over mechanical endoscopy (shorter acquisition times, increased patient comfort, higher cost-effectiveness, no complications of endoluminal instrumentation, field-of-view extending beyond the surface). In addition, simulated endoscopy can be used in virtual spaces that can not at all, or only after violation of normal anatomical structures, be reached by mechanical (endo)-scopy.

From a clinical perspective, there is a demand in community hospitals to make an environment available, suitable for the interactive rendering and interactive matching of, and switching between the above described data sets with an

5 Endoscopy is a diagnostic procedure where an instrument is used to visualize the interior of a hollow organ.

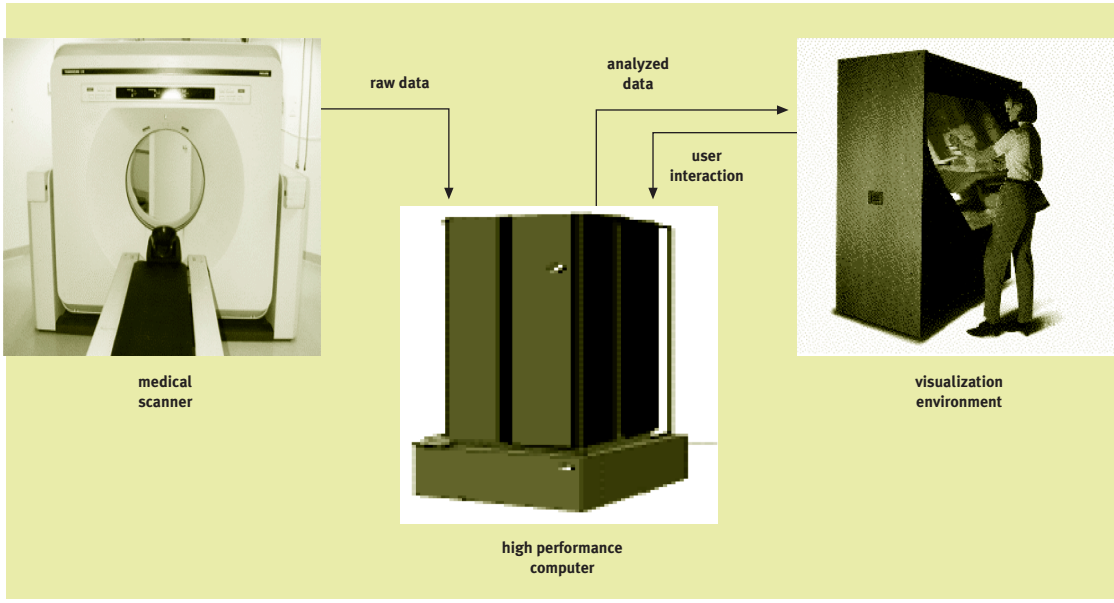


Figure 6
The VRE environment allows medical data from hospitals to be pre-processed on HPC systems for 3D visualization. High speed networking initiatives such as the GigaPort project [gigaport homepage, 2000] allow hospitals to make interactive use of HPC visualization techniques for patient diagnostics.

emphasis on simulated endoscopy. The VRE environment provides various such methods for the visualization of medical scans, including volume rendering using SGI's Volumizer [Volumizer homepage, 2000], surface rendering using VTK [Schroeder, 1997] and OpenGL [OpenGL homepage, 2000], interactive clipping and surface mapping techniques. Mechanisms have been added that allow the VRE environment to be run in a CAVE or on an ImmersaDesk. The ImmersaDesk allows the VRE environment to be used in the radiology department. Shown in Figure 7 is an isosurface representation of the abdominal aorta from an MRA scan. A geometric probing system (GEOPROVE, see [Belleman, 1999]) is used to perform measurements on this representation.

VRE+

VRE+ extends VRE with methods that allow dynamic exploration. Various methods are added to visualize the results of a simulation, while it is running. An intelligent agent system is integrated that constantly monitors a user's actions and provides feedback to the user. Currently, we have implemented a speech recognition agent, which enables users to control the environment using hands and voice simultaneously. A second agent monitors the position of the user, when using GEOPROVE and provides feedback on the accuracy of the measurements. Another agent monitors the state of the simulation environment and provides feedback on the state of the simulation.

For the planning part, the VRE+ environment is extended with means to 'draw' a surgical procedure using a 'grid editor', as described in the section on grid generation and editing.

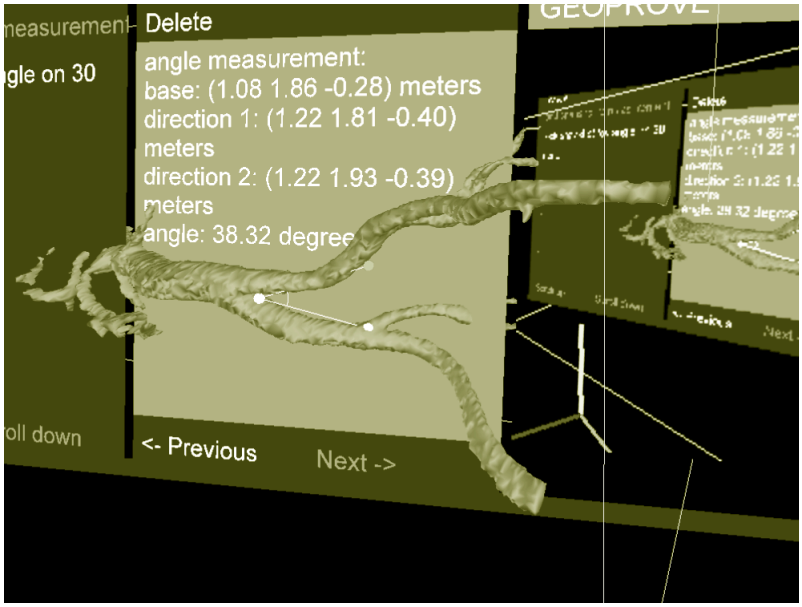


Figure 7
A snapshot of the VRE environment running in a CAVE. An isosurface representation of an abdominal aorta is shown obtained from an MRA scan. The panel shows the GEOPROVE environment, which allows measurements and annotations (such as the virtual 'snapshot' on the right) to be made from within the environment.

Fluid flow simulation: the lattice-Boltzmann method

The lattice-Boltzmann method (LBM) is a mesoscopic approach for simulating fluid flow based on the kinetic Boltzmann equation [Chen, 1998]. In this method fluid is modeled by particles moving on a regular lattice. At each time step, particles propagate to neighboring lattice points and re-distribute their velocities in a local collision phase. This inherent spatial and temporal locality of the update rules makes this method ideal for parallel computing [Kandhai, 1998]. During recent years, LBM has been successfully used for simulating many complex fluid-dynamical problems, such as suspension flows, multi-phase flows, and fluid flow in porous media [Koponen, 1998]. All these problems are quite difficult to simulate by conventional methods [Kandhai, 1999a; Kandhai, 1999b]. The data structures required by LBM (Cartesian grids) bear a great resemblance to the grids that come out of CT and MRI scanners. As a result, the amount of preprocessing can be kept to a minimum, which reduces the risk of introducing errors due to data structure conversions. In addition, LBM has the benefit over other fluid flow simulation methods that flow around (or through) irregular geometries (like a vascular structure) can be simulated relatively easily. Yet another advantage of LBM is the possibility to calculate the shear stress on the arteries directly from the densities of the particle distributions [Artoli, 2000]. This may be beneficial in cases where we want to interfere with the simulation while the velocity and the stress field are still developing, thus supporting fast data updating given a proposed change in simulation parameters from the interaction modules.

Lattice-Boltzmann grid generation and editing

As mentioned earlier, the basic structure of the grids used in LBM bear great resemblance to the medical scans obtained from a patient. To convert the medical scans into LBM grids, the raw data from the medical scanner is first segmented so that only the arterial structures of interest remain in the data set (see also Figure 8). The contrast fluid injected into the patient in a MRA scan provides sufficient contrast in the vascular structures to do this quite efficiently. The segmented data set is then converted into a grid that can be used in LBM; boundary nodes, inlet nodes and outlet nodes are added to the grid using a variety of image processing techniques.

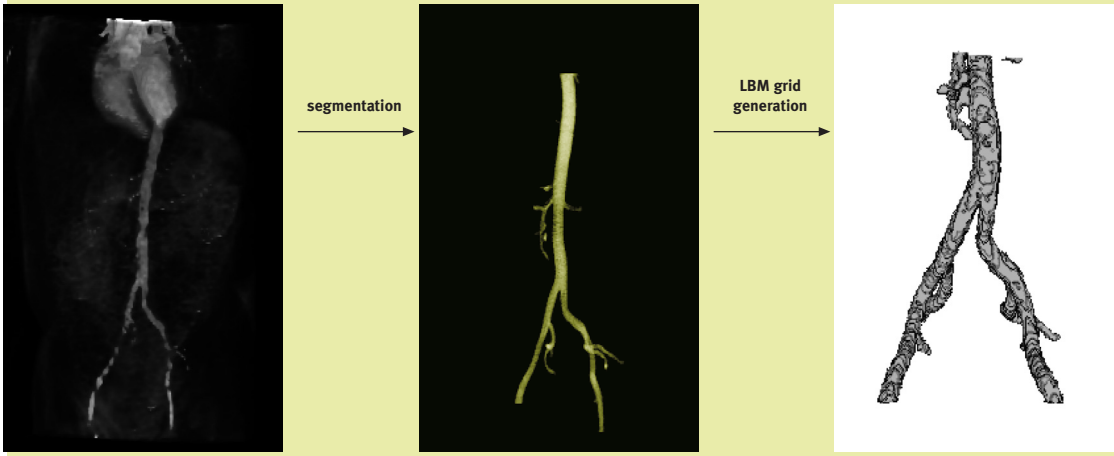


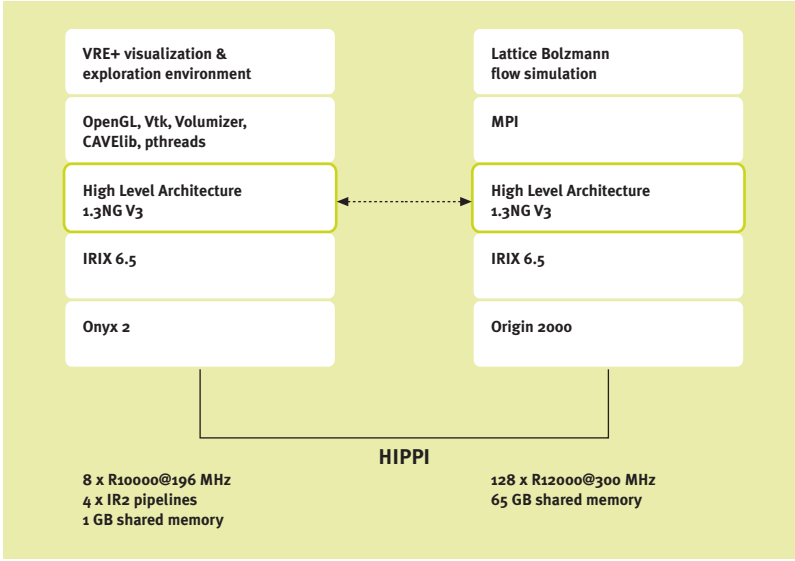
Figure 8
LBM grids are generated from raw medical scans through a combination of segmentation and image processing techniques.

A surgical procedure is simulated through the use of a 3D grid editor. This system allows a user to interactively add and or delete areas in the LBM grid corresponding to the procedure that is simulated. Similar grid generation techniques as described above are used to ensure the grids comply with the demands imposed by LBM.

Middleware

The different components involved in our interactive simulation system are shown in Figure 9. As can be seen from this figure, the visualization and exploration system runs on a different system (a CAVE) than the simulation system (a massively parallel Origin 2000). HLA is used as a middleware layer to connect these components together. By using HLA, the different components can run asynchronously, while spatial and temporal effects as described in the Section on time management can be controlled. In addition, HLA provides attribute ownership management and does efficient data distribution between heterogeneous (if needed) systems.

Figure 9
The visualization and exploration environment (on the left, running in a CAVE) and the simulation system communicate via the HLA.



Discussion and future work

We have presented our views on dynamic exploration environments that support distributed interactive simulation. We have provided an overview on the requirements of such an environment and the issues involved in its construction. We have described how the HLA offers all requirements that are needed in its basic architecture. The case described in the final section is presented as a prototypical case study to validate these assumptions.

Preliminary measurements on the test case environment show that HLA is a suitable architecture to build a relatively efficient interactive and distributed simulation environment. Compared to raw network performance, communication overhead and delays imposed by HLA are acceptable. Implementing a HLA federation, however, requires a substantial effort, but is mostly due to the lack of proper software development tools.

The performance of the test case simulation environment will be validated through a comparison of fluid flow simulation results and the results of other simulation methods as well as in vivo measurements of blood flow through phantom structures and pre- and post-operative MRA scans.

Acknowledgements

This research is funded through grant 612-21-103 from the Netherlands Organization for Scientific Research (NWO). We are also greatly indebted to Charles A. Taylor (Department of Mechanical Engineering, Stanford University) for his insightful and inspiring discussions and for allowing us to use his data sets, Sean A. Spicer (Department of Mechanical Engineering, Stanford

University) for providing his Volumizer Convenience Classes and enabling them for use in the CAVE, Silicon Graphics Inc. for their patience in answering all our questions and fixing bugs in the course of this research, Drona Kandhai (Section Computational Science, The Universiteit van Amsterdam) for his work on the Lattice-Boltzmann fluid simulation environment and Zhiming Zhao (Section Computational Science, The Universiteit van Amsterdam) for his work on HLA and IA's. Finally, we would like to thank Eva Rombouts for her medical input and Alfons Hoekstra for his helpful remarks, while reading this document.

References

- Academic Computing Services Amsterdam (SARA). (1998). Amsterdam, The Netherlands. SARA - CAVE Homepage. http://www.sara.nl/hec/vr/cave/
- Artoli, A.M., D. Kandhai, A.G. Hoekstra, P.M.A. Sloot. (2000). Accuracy of Shear Stress Calculations in the Lattice Boltzmann Method. Accepted for the 9th International Conference on Discrete Simulation of Fluid Dynamics
- Basdogan, C., H. Chih-Hao, M.A. Srinivasan. (1999). Simulation of Tissue Cutting and Bleeding for Laparoscopic Surgery Using Auxiliary Surfaces. In: J.D. Westwood, H.M. Homan, R.A. Robb, D. Stredney. (eds.). (1999). Medicine Meets Virtual Reality. pp38-44. IOS Press, Amsterdam
- Belleman, R.G., J.A. Kaandorp, P.M.A. Sloot. (1998). A Virtual Environment for the Exploration of Diffusion and Flow Phenomena in Complex Geometries. Future Generation Computer Systems **14** (3-4):209-214
- Belleman, R.G., J.A. Kaandorp, D. Dijkman, P.M.A. Sloot. (1999). GEOPROVE: Geometric Probes for Virtual Environments. In: P.M.A. Sloot, M. Bubak, A. Hoekstra, L.O. Hertzberger. (eds.). High Performance Computing and Networking (HPCN'99). pp817-827, Amsterdam, The Netherlands. Springer Verlag
- Bockholt, U., U. Ecke, W. Muller, G. Voss. (1999). Realtime Simulation of Tissue Deformation for the Nasal Endoscopy Simulator (NES). In: J.D. Westwood, H.M. Homan, R.A. Robb, D. Stredney. (eds.). Medicine Meets Virtual Reality. pp74-75. IOS Press, Amsterdam
- Bryson, S. (1996a). Virtual Reality in Scientific Visualization. Communications of the ACM **39** (5):62-71
- Bryson, S., S. Johan. (1996b). Time Management, Simultaneity and Time-Critical Computation in Interactive Unsteady Visualization Environments. Proceedings of Visualization '96. p255. IEEE Computer Science Press, Los Alamitos, CA
- Chen, J.X., D. Rine, H.D. Simon. (1996). Advancing Interactive Visualization and Computational Steering. IEEE Computational Science & Engineering, pp13-17
- Chen, S., G.D. Doolen. (1998). Lattice Boltzmann Method for Fluid Flows. Annu. Rev. Fluid Mech. **30**:329

- Cruz-Neira, C., D.J. Sandin, T.A. DeFanti. (1993). Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. SIGGRAPH '93 Computer Graphics Conference. pp135-142. ACM SIGGRAPH
- Defense Modeling and Simulation Office (DMSO). (1999). Department of Defense, US. High Level Architecture Run Time Infrastructure Programmer's Guide (1.3 version 7). http://hla.dmso.mil/
- Defense Modeling and Simulation Office (DMSO). (1999). High Level Architecture (HLA) homepage. http://hla.dmso.mil/
- Durnford, L. (1999). Virtual Reality: More than just a Game. Radio Netherlands Wereldomroep. http://www.rnw.nl/science/html/virtualreality990514.html
- Elvins, T. (1997). Virtually Lost in Virtual Worlds — Wayfinding without a Cognitive Map. Computer Graphics. http://www.sdsc.edu/~todd/
- Foster, I., C. Kesselman. (1997). Globus: A Metacomputing Infrastructure Toolkit. International Journal Supercomputer Applications **11** (2):115-128
- Hayes-Roth, B. (1995). An Architecture for Adaptive Intelligent Systems. Artificial Intelligence. Special Issue on Agents and Interactivity **72**:329-365
- IBM Corporation, Armonk, NY. (1991). Data Explorer Reference Manual
- Johnson, Ch.R., S.G. Parker. Applications in Computational Medicine Using SCIRun: A Computational Steering Programming Environment. In: H.W. Meuer. (ed.). Supercomputer '95. pp2-19
- Kaandorp, J.A. (ed.). (1998). Future Generation Computer Systems. Special Double Issue on Virtual Reality in Industry and Research **14** (3-4). Elsevier Science
- Kandhai, D., A. Koponen, A.G. Hoekstra, M. Kataja, J. Timonen, P.M.A. Sloot. (1998). Lattice Boltzmann Hydrodynamics on Parallel Systems. Computer Physics Communications
- Kandhai, D. (1999a). Large Scale Lattice-Boltzmann Simulations (Computational Methods and Applications). PhD Thesis. The Universiteit van Amsterdam, Amsterdam, The Netherlands
- Kandhai, D., D. Vidal, A. Hoekstra, H. Hoefsloot, P. Iedema, P.M.A. Sloot. (1999b). Lattice-Boltzmann and Finite Element Simulations of Fluid Flow in a SMRX Mixer. Int. J. Numer. Meth. Fluids **31**:1019-1033
- Koponen, A., D. Kandhai, E. Hellen, M. Alava, A. Hoekstra, M. Kataja, K. Niskanen, P. Sloot, J. Timonen. (1998). Permeability of Three-Dimensional Random Fiber Webs. Physical Review Letters **0** (4):716-719
- Ku, D.N. (1997). Blood Flow in Arteries. Annu. Rev. Fluid Mech. **29**:399-434
- Liere, R. van, J.D. Mulder, J.J. van Wijk. (1996). Computational Steering. In: H. Liddell, A. Colbrook, B. Hertzberger, P. Sloot. (eds.). High-Performance Computing and Networking. pp696-702. Springer Verlag
- Mulder, J.D., J.J. van Wijk. (1995). 3D Computational Steering with Parameterized Geometric Objects. In: G.M. Nielson, D. Silver. (eds.). IEEE

Visualization '95, pp304-312. IEEE CS
- Overeinder, B.J., P.M.A. Sloot. (1999). Extensions to Time Warp Parallel Simulation for Spatially Decomposed Applications. In: D. Al-Dabass, R. Cheng. (eds.). Proceedings of the Fourth United Kingdom Simulation Society Conference (UKSim 99). pp67-73. Cambridge, UK
- Parker, S.G., Ch.R. Johnson. (1995). SCIRun: A Scientific Programming Environment for Computational Steering. Supercomputing '95
- Roy, T.M., C. Cruz-Neira, T.A. DeFanti, D.J. Sandin. (1995). Steering a High Performance Computing Application from a Virtual Environment. Presence: Teleoperators and Virtual Environments **4** (2):121-129
- Schroeder, W., K. Martin, B. Lorensen. (1997). The Visualization Toolkit, an Object-Oriented Approach to 3D Graphics. 2nd edition. Prentice Hall, Upper Saddle River, NJ
- Silicon Graphics Inc. Software Products. (2000). OpenGL homepage. http://www.sgi.com/software/opengl/
- Silicon Graphics Inc. Software Products. (2000). Volumizer homepage. http://www.sgi.com/software/volumizer/
- Surfnet. Gigaport homepage. (2000). http://www.gigaport.nl/en index.html
- Taylor, Ch.A., Th.J.R. Hughes, Ch.K. Zarins. (1998). Finite Element Modeling of Three-Dimensional Pulsatile Flow in the Abdominal Aorta: Relevance to Atherosclerosis. Annals of Biomedical Engineering **26**:975-987
- Taylor, V.E., J. Chen, T.L. Disz, M.E. Papka, R. Stevens. (1996). Interactive Virtual Reality in Simulations: Exploring Lag Time. IEEE Computational Science and Engineering. pp46-54
- The Numerical Algorithms Group Ltd. (1998). Oxford, UK. Iris Explorer User's Guide
- Upson, C., T. Faulhaber, jr., D. Kamins. (et al.). (1989). The Application Visualization System: a Computational Environment for Scientific Visualization. IEEE Computer Graphics and Applications **9** (4):30-42
- Versweyveld, L. (1998). Exploring the Medical Applications of Virtual Reality Techniques in the Dutch CAVE. Virtual Medical Worlds. http://www.hoise.com/vmw/articles/LV-VM-04-98-13.html
- Yucel, E.K., Ch.M. Anderson, R.R. Edelman, Th.M. Grist, R.A. Baum, W.J. Manning, A. Culebras, W. Pearce. (1999). Magnetic Resonance Angiography. Update on Applications for Extracranial Arteries). Circulation **100**:2284-2301