Handbook of Statistics, Vol. 24
ISSN: 0169-7161

DOI 10.1016/S0169-7161(04)24017-2

Data Visualization and Virtual Reality

Jim X. Chen

1. Introduction

We present a survey of basic principles and advanced technologies in data visualization, with a focus on Virtual Reality (VR) hardware and software. In order to understand visualization and VR, we should understand computer graphics first. Most data visualization today relies on computer graphics to present data in innovative ways, so that obscured abstract or complex information become obvious and easy to comprehend through images and/or animations. VR, also called VE (Virtual Environment) in some applications, is an extension to computer graphics to include other human senses, in addition to eyes, that perceive data, phenomena, or simulated virtual world from different perspectives. VR extends the 3D graphics world to include stereoscopic, acoustic, haptic, tactile, and other kinds of feedbacks to create a sense of immersion.

In the rest of the chapter, we first concisely introduce principles of computer graphics and graphics software tools (Chen, 2002, Foley et al., 1996), then provide some data visualization technologies and a few examples (Chen and Wang, 2001, Wang et al., 2002, Wegman and Carr, 1993, Wegman and Luo, 1997), after that discuss VR and software (Burdea and Coiffet, 2003, Stanney, 2002, Vince, 1995, Vince, 1998, Wegman, 2000, Wegman and Symanzik, 2002), and finally list some applications that exploit an integration of visualization and VR.

2. Computer graphics

A graphics *display device* is a drawing board with an array of fine points called pixels. At the bottom of a graphics system is a mechanism named *magic pen* here, which can be moved to a specific pixel position, and tune the pixel into a specific color (an RGB vector value). This magic pen can be controlled directly by hand through an input device (mouse, keyboard, etc.) as a simple paintbrush. In this case, we can draw whatever we imagine, but it has nothing to do with computer graphics and data visualization. Computer graphics, or simply graphics, is more about controlling this magic pen automatically through programming to generate images or animation: given input data (geometric objects, computational results, etc.), the application program will move

the magic pen to turn the abstract data into objects in a 2D or 3D image. Here, we will explain what we need to provide to a graphics system and how a graphics system works in general.

2.1. Shape

In computer graphics, unlike drawing by hand, we need to provide accurate description of the shape of the object to be drawn: a list of vertices or a geometric description (such as a specific curve equation) which can be used to find all the points of the object. Therefore, instead of just having a picture, we have a *model* that describes the object as initial data saved in the memory. A *graphics library* or *package* provides a set of graphics commands or output subroutines. Graphics commands can specify primitive geometric models that will be digitized and displayed. Here "primitive" means that only certain simple shapes (such as points, lines, polygons) can be accepted by a graphics library. A very complex shape needs an application program to assemble small pieces of simple shapes. As you know, we have the magic pen that can draw a pixel. If we can draw a pixel (point), we can draw a line, a polygon, a curve, a block, a building, an airplane, etc. A general application program could be included into a graphics library as a command to draw a complex shape.

A graphics system digitizes a specific model into a set of discrete color points saved in a piece of memory called *frame buffer*. This digitization process is called *scan-conversion*. The color points in the frame buffer will be sent to the corresponding pixels in the display device by a piece of hardware called *video controller*. Therefore, whatever in the frame buffer corresponds to an image on the screen. The display device indicates both the frame buffer and the screen. The application program accepts user input, manipulates the model (create, store, retrieve, and modify the descriptions), and produces the picture through the graphics system. The display device is also a window for us to manipulate the image as well as the model behind the image through the application program. Most of the programmer's task concerns creating and editing the model, and handling user interaction in a programming language with a graphics library.

2.2. Transformation

After we have the model (description of the object), we may move it around or even transform the shape of the model into a completely different object. Therefore, we need to specify the rotation axis and angle, translation vector, scaling vector, or other manipulations to the model. The ordinary *geometric transformation* is a process of mathematical manipulations of each vertices of a model through matrix multiplication, and the graphics system then displays the final transformed model. The transformation can be predefined, such as moving along a planed trajectory, or interactive depending on user input. The transformation can be permanent, which means that the coordinates of the vertices are changed and we have a new model replacing the original one, or just temporarily. In many cases a model is transformed in order to display it at a different position/orientation, and the graphics system discard the transformed model after scan-conversion. Sometimes, all the vertices of a model go through the same transformation

and the shape of the model is preserved; sometimes, different vertices go through different transformations, and the shape is dynamic.

In a movie theater, motion is achieved by taking a sequence of pictures and projecting them at 24 frames per second on the screen. In a graphics system, as we assumed, the magic pen moves at lightning speed. Therefore, we can first draw an object and keep it on display for a short period of time, then erase it and redraw the object after a small step of transformation (rotation, translation, etc.), and repeat this process. The resulting effect is that the object is animated on display. Of course, in order to achieve animation for complex objects, we need not only a fast pen, but also other high performance algorithms and hardware that carry out many graphics functions (including transformation and scan-conversion) efficiently. In order to achieve smooth transformation, two frame buffers are used in the hardware. While one frame buffer is used for displaying, the other is used for scan-conversion, and then they are swapped so the one that was displaying is used for scan-conversion, and the one that was used for scan-conversion is now for displaying. This hardware mechanism allows parallelism and smooth animation.

2.3. Viewing

The display device is a 2D window, and our model may be in 3D. When we specify a model and its transformations in a coordinate system, we need to consider the relationship between this coordinate system and the coordinates of the display device (or its corresponding frame buffer.) Therefore, we need a *viewing* transformation, the mapping of a part of the model's coordinate scene to the display device coordinates. We need to specify a viewing volume, which determines a projection method (*parallel* or *perspective*) – how 3D model is projected into 2D. The viewing volume for parallel projection is like a box. The result of parallel projection is a less realistic view, but can be used for exact measurements. The viewing volume for perspective projection is like a truncated pyramid, and the result looks more realistic in many cases, but does not preserve sizes on display – objects further away are smaller.

Through the graphics system, the model is clipped against the 3D viewing volume and then projected into a 2D plane corresponding to the display window. This transformation process is analogous to taking a photograph with a camera. The object in the outside world has its own coordinate system. Our film in the camera has its own coordinate system. By pointing and adjusting the zoom, we have specified a viewing volume and projection method. If we aim at objects very far away, we can consider the projection is parallel, otherwise perspective.

2.4. Color and lighting

We can specify one color for a model, or different colors for different vertices. The colors of the pixels between the vertices are interpolated by the graphics system.

On the other hand, we can specify color indirectly by setting light sources of different colors at different positions in the environment and material properties, so the model will be lit accordingly. When we specify light sources, we need also specify the material properties (*ambient*, *diffuse*, and *specular* parameters) of the model, so the model can have overall brightness, dull reflections, and shiny spots. The graphics system achieves

lighting (namely, *illumination* or *shading*) by calculating the color, or more precisely, the RGB vector of each individual pixel in the model. It uses empirical methods to integrate the material properties, the normal of the pixel, the position of the viewer, and the colors and positions of the light sources. The normal of a pixel tells the system which direction the pixel is facing. It can be calculated/interpolated by the normals of the surrounding vertices of the pixel in the model. It can be either provided or calculated from its corresponding surface equation.

A more expensive method, which is generally not provided in a low-level graphics library, is to calculate lightness by *ray-tracing*. A ray is sent from the view point through a point on the projection plane to the scene to calculate the recursive intersections with objects, bounces with reflections, penetrations with refractions, and finally the color of the point by accumulating all fractions of intensity values from bottom up. The drawback is that the method is time consuming.

2.5. Texture mapping

When calculating the color of a pixel, we can also use or integrate a color retrieved from an image. To do this, we need to provide a piece of image (array of RGB values) called texture. Therefore, *texture mapping* is to tile a surface of a model with existing texture. At each rendered pixel, selected texture pixels (*texels*) are used either to substitute or scale one or more of the surface's material properties, such as its diffuse color components. Textures are generated from real images scanned into memory or by any other means.

2.6. Transparency

When we have transparent objects, or even want to visualize the contents of an opaque object, we need to specify the *transmission coefficients* for overlapping objects. A final pixel's R, G, or B value depends on the blending of the corresponding R, G, or B values of the pixels in the overlapping objects through the transmission coefficients: $I_\lambda = \alpha_1 I_{\lambda 1} + \alpha_2 I_{\lambda 2}$, where λ is R, G, or B; $I_{\lambda 1}$ the color component of the pixel in the first object, α_1 the transmission coefficient of the pixel in the first object, and similarly for the second object.

2.7. Graphics libraries

A graphics system includes a graphics library and the corresponding hardware. A graphics library includes most of the basic components we have described in this section. Some of the functions are more or less implemented in hardware, otherwise they would be very slow. Some of the functions requires us to provide input, as a list of vertices for drawing a polygon, and others we do not, as removing hidden surfaces. In a graphics library, like controlling the pen by hand, we may draw pixels directly into frame buffer for display without going through transformation and viewing. In general, we provide vertices for primitives (points, lines, or polygons), and they all go through similar operations: transformation, viewing, lighting, scan-conversion, etc. OpenGL is currently the most popular low-level graphics library. Direct3D and PHIGS are two other well-known graphics libraries.

3. Graphics software tools

A low-level graphics library or package is a software interface to graphics hardware, namely the application programmer's interface (API). All graphics tools or applications are built on top of a low-level graphics library, such as visualization, rendering, and VR tools (http://cs.gmu.edu/~jchen/graphics/book/index.html), which we discuss in more detail in this section (Chen, 2002). A taxonomy of graphics tools can be found at http://cs.gmu.edu/~jchen/graphics/book/index_tools_catagories.html.

3.1. Visualization

Visualization employs graphics to make pictures that give us insight into certain abstract data and symbols. The pictures may directly portray the description of the data, or completely present the content of the data in an innovative form. Many visualization tools have multiple and overlapping visualization functions. Some tools include capabilities of interactive modeling, animation, simulation, and graphical user interface construction as well. In the following, we briefly introduce several visualization tools.

AVS/Express, IRIS Explorer, Data Explorer, MATLAB, PV-WAVE, Khoros, and Vtk are multiple purpose visualization commercial products that satisfy most of the visualization needs. *AVS/Express* has applications in many scientific areas, including engineering analysis, CFD, medical imaging, and GIS (Geographic Information Systems). It is built on top of OpenGL and runs on multiple platforms. *IRIS Explorer* includes visual programming environment for 3D data visualization, animation and manipulation. IRIS Explorer modules can be plugged together, which enable users to interactively analyze collections of data and visualize the results. IRIS Explorer is build on top of *OpenInventor*, an interactive 3D object scene management, manipulation, and animation tool. OpenInventor has been used as the basis for the emerging Virtual Reality Modeling Language (VRML). The rendering engine for IRIS Explorer and OpenInventor are OpenGL. IBM's *Data Explorer* (*DX*) is a general-purpose software package for data visualization and analysis. OpenDX is the open source software version of the DX Product. DX is build on top of OpenGL and runs on multiple platforms. *MATLAB* was originally developed to provide easy access to matrix software. Today, it is a powerful simulation and visualization tool used in a variety of application areas including signal and image processing, control system design, financial engineering, and medical research. *PV-WAVE* integrates charting, volume visualization, image processing, advanced numerical analysis, and many other functions. *Khoros* is a software integration, simulation, and visual programming environment that includes image processing and visualization. *Vtk* is a graphics tool that supports a variety of visualization and modeling functions on multiple platforms. In Vtk, applications can be written directly in C++ or in Tcl (an interpretive language).

Volumizer, 3DVIEWNIX, ANALYZE, and VolVis are 3D imaging and volume rendering tools. *Volume rendering* is a method of extracting meaningful information from a set of volumetric data. For example, a sequence of 2D image slices of human body can be reconstructed into a 3D volume model and visualized for diagnostic purposes or for planning of treatment or surgery. NIH's Visible Human Project

(http://www.nlm.nih.gov/research/visible/visible_human.html) creates anatomically detailed 3D representations of the human body. The project includes effort of several universities and results in many imaging tools.

StarCD, FAST, pV3, FIELDVIEW, EnSight, and Visual3 are CFD (Computational Fluid Dynamics) visualization tools. Fluid flow is a rich area for visualization applications. Many CFD tools integrate interactive visualization with scientific computation of turbulence or plasmas for the design of new wings or jet nozzles, the prediction of atmosphere and ocean activities, and the understanding of material behaviors.

NCAR, Vis5D, FERRET, Gnuplot, and SciAn are software tools for visual presentation and examination of data sets from physical and natural sciences, often requiring the integration of terabyte or gigabyte distributed scientific databases with visualization. The integration of multi-disciplinary data and information (e.g. atmospheric, oceanographic, and geographic) into visualization systems will help and support cross-disciplinary explorations and communications.

3.2. Modeling and rendering

Modeling is a process of constructing a virtual 3D graphics object (namely, computer model, or simply model) from a real object or an imaginary entity. Creating graphics models require a significant amount of time and effort. Modeling tools make creating and constructing complex 3D models easy and simple. A graphics model includes geometrical descriptions (particles, vertices, polygons, etc.) as well as associated graphics attributes (colors, shadings, transparencies, materials, etc.), which can be saved in a file using certain standard (3D model) formats. Modeling tools help create virtual objects and environments for CAD (computer-aided design), visualization, education, training, and entertainment. *MultigenPro* is a powerful modeling tool for 3D objects and terrain generation/editing. *AutoCAD* and *MicroStation* are popular for 2D/3D mechanical designing and drawing. *Rhino3D* is for free-form curve surface objects.

Rendering is a process of creating images from graphics models (Watt and Watt, 1992). 3D graphics models are saved in computer memory or hard-disk files. The term *rasterization* and *scan-conversion* are used to refer to low-level image generation or drawing. All modeling tools provide certain drawing capabilities to visualize the models generated. However, in addition to simply drawing (scan-converting) geometric objects, rendering tools often include lighting, shading, texture mapping, color blending, ray-tracing, radiosity, and other advanced graphics capabilities. For example, *RenderMan* Toolkit includes photo-realistic modeling and rendering of particle system, hair, and many other objects with advanced graphics functions such as ray-tracing, volume display, motion blur, depth-of-field, and so forth. Some successful rendering tools were free (originally developed by excellent researchers at their earlier career or school years), such as POVRay, LightScape, Rayshade, Radiance, and BMRT. *POVRay* is a popular ray-tracing package across multiple platforms that provides a set of geometric primitives and many surface and texture effects. *LightScape* employs radiosity and ray-tracing to produce realistic digital images and scenes. *Rayshade* is an extensible system for creating ray-traced images that includes a rich set of primitives, CSG (constructive solid geometry) functions, and texture tools. *Radiance* is a rendering package

for the analysis and visualization of lighting in design. It is employed by architects and engineers to predict illumination, visual quality and appearance of design spaces, and by researchers to evaluate new lighting technologies. *BMRT* (Blue Moon Rendering Tools) is a RenderMan-compliant ray-tracing and radiosity rendering package. The package contains visual tools to help users create RenderMan Input Bytestream (RIB) input files.

Many powerful graphics tools include modeling, rendering, animation, and other functions into one package, such as Alias|Wavefront's Studio series and Maya, SoftImage, 3DStudioMax, LightWave, and TrueSpace. It takes serious course training to use these tools. Alias|Wavefront's *Studio* series provide extensive tools for industrial design, automotive styling, and technical surfacing. Its *Maya* is a powerful and productive 3D software for character animation that has been used to create visual effects in some of the hottest recent film releases, including A Bug's Life and Titanic. *SoftImage3D* provides advanced modeling and animation features such as NURBS, skin, and particle system that are excellent for special effects and have been employed in many computer games and films, including stunning animations in Deep Impact and Airforce One. *3DStudioMax* is a popular 3D modeling, animation, and rendering package on Window 95/NT platform for game development. Its open plugin architecture makes it an idea platform for third party developers. *LightWave* is another powerful tool that has been successfully used in many TV feature movies, games, and TV commercials. *TrueSpace* is another popular and powerful 3D modeling, animation, and rendering package on Win95/NT platforms.

3.3. Animation and simulation

In a movie theater, animation is achieved by taking a sequence of pictures, and then projecting them at 24 frames per second on the screen. Videotape is shown at 30 frames per second. Computer animation is achieved by refreshing the screen display with a sequence of images at 60–100 frames per second. Of course, we can use fewer than 60 images to refresh 60 frames of screen in a second, so that some images will be used to refresh the display more than one time. The fewer the number of different images displayed in a second, the jerkier the animation will be. Keyframe animation is achieved by pre-calculate keyframe images and in-between images, which may take significant amount of time, and then display (play-back) the sequence of generated images in real time. Keyframe animation is often used in visual effects in films and TV commercials, which no interactions or unpredictable changes are necessary. Interactive animation, on the other hand, is achieved by calculating, generating, and displaying the images at the same time on the fly. When we talk about real-time animation, we mean the virtual animation happens in the same time frames as in real world behavior. However, for graphics researchers, real-time animation often simply implies the animation is smooth or interactive. Real-time animation is often used in virtual environment for education, training, and 3D games. Many modeling and rendering tools are also animation tools, which are often associated with simulation.

Simulation, on the other hand, often means a process of generating certain natural phenomena through scientific computation. The results of the simulations may be large

datasets of atomic activities (positions, velocities, pressures and other parameters of atoms) or fluid behaviors (volume of vectors and pressures). Computer simulation allows scientists to generate the atomic behavior of certain nanostructured materials for understanding material structure and durability and to find new compounds with superior quality (Nakano et al., 1998). Simulation integrated with visualization can help pilots to learn to fly and aid automobile designers to test the integrity of the passenger compartment driving crashes. For many computational scientists, simulation may not be related to any visualization at all. However, for many graphics researchers, simulation often means simply animation. Today, graphical simulation, or simply simulation, is an animation of certain process or behavior that are generated often through scientific computation and modeling. Here we emphasize an integration of simulation and animation – the simulated results are used to generate graphics model and control animation behaviors. It is far easier, cheaper, and safer to experiment with a model through simulation than with a real entity. In fact, in many situations, such as training space-shuttle pilots and studying molecular dynamics, modeling and simulation are the only feasible method to achieve the goals. *Real-time simulation* is an overloaded term. To computational scientists, it often means the simulation time is the actual time in which the physical process (under simulation) should occur. In automatic control, it means the output response time is fast enough for automatic feedback and control. In graphics, it often means that the simulation is animated at an interactive rate of our perception. The emphasis in graphics is more on responsiveness and smooth animation rather than strictly accurate timing of the physical process. In many simulations for training applications, the emphasis is on generating realistic behavior for interactive training environment rather than strictly scientific or physical computation.

IRIS Performer is a toolkit for real-time graphics simulation applications. It simplifies development of complex applications such as visual simulation, flight simulation, simulation-based design, virtual reality, interactive entertainment, broadcast video, CAD, and architectural walk-through. *Vega* is MultiGen-Paradigm's software environment for real-time visual and audio simulation, virtual reality, and general visualization applications. It provides the basis for building, editing, and running sophisticated applications quickly and easily. *20-sim* is a modeling and simulation program for electrical, mechanical, and hydraulic systems or any combination of these systems. *VisSim/Comm* is a Windows-based modeling and simulation program for end-to-end communication systems at the signal or physical level. It provides solutions for analog, digital, and mixed-mode communication system designs. *SIMUL8* is a visual discrete event simulation tool. It provides performance measures and insights into how machines and people will perform in different combinations. *Mathematica* is an integrated environment that provides technical computing, simulation, and communication. Its numeric and symbolic computation abilities, graphical simulation, and intuitive programming language are combined with a full-featured document processing system. As we discussed earlier *MATLAB*, *Khoros*, and many other tools contains modeling and simulation functions.

3.4. File format converters

Currently, 3D model file formats and scene description methods are numerous and different. Different modeling tools create models with a variety of attributes and

parameters, and use their own descriptions and formats. It happens that we are unable to use certain models because we cannot extract certain information from the files. There are 3D model and scene file format converting tools available (Chen and Yang, 2000), such as *PolyTran*, *Crossroads*, *3DWin*, *InterChange*, *Amapi3D*, *PolyForm*, *VIEW3D*, and *Materialize3D*. Some attributes and parameters unique to certain formats will be lost or omitted for simplicity in the conversions.

3.5. Graphics user interfaces

Graphical user interface tools, such as *Motif*, *Java AWT*, *Microsoft Visual C++ MFC/Studio*, *XForms*, and *GLUI*, are another category of graphics tools. Computer graphics covers a wide range of applications and software tools. The list of software tools is growing exponentially when we try to cover more. We list all the software tools covered in this chapter at http://cs.gmu.edu/graphics/.

4. Data visualization

Computer graphics provides the basic functions for generating complex images from abstract data. Visualization employs graphics to give us insight into abstract data. Apparently, a large variety of techniques and research directions exist in data visualization. Choosing a specific technique is mostly determined by the type and the dimension of the data to be visualized. The generally discussed research areas include volume visualization for volumetric data, vector fields and flow visualization for computational fluid dynamics, large data-set visualization for physical and natural sciences, information visualization for text libraries, and statistical visualization for statistical data analysis and understanding. Many visualization techniques deal with multidimensional multivariate data sets.

4.1. Data type

A visualization technique is applicable to data of certain types (discrete, continual, point, scalar, or vector) and dimensions (1D, 2D, 3D, and multiple: *N*D). *Scatter Data* represent data as discrete points on a line (1D), plane (2D), or in space (3D). We may use different colors, shapes, sizes, and other attributes to represent the points in high dimensions beyond 3D, or use a function or a representation to transform the high-dimensional data into 2D/3D. *Scalar Data* have scalar values in addition to dimension values. The scalar value is actually a special additional dimension that we pay more attention. 2D diagrams like histograms, bar charts, or pie charts are 1D scalar data visualization methods. Both histograms and bar charts have one coordinate as the dimension scale and another as the value scale. Histograms usually have scalar values in confined ranges, while bar charts do not carry this information. Pie charts use a slice area in a pie to represent a percentage. 2D contours (iso-lines in a map) of constant values, 2D images (pixels of x–y points and color values), and 3D surfaces (pixels of x–y points and height values) are 2D scalar data visualization methods. Volume and iso-surface rendering methods are for 3D scalar data. A *voxel* (volume pixel) is a 3D scalar data with

(x, y, z) coordinates and an intensity or color value. *Vector Data* include directions in addition to scalar and dimension values. We use line segments, arrows, streamlines, and animations to present the directions.

4.2. Volumetric data – volume rendering

Volume rendering or visualization is a method of extracting meaningful information from a set of 2D scalar data. A sequence of 2D image slices of human body can be reconstructed into a 3D volume model and visualized for diagnostic purposes or for planning of treatment or surgery. For example, a set of volumetric data such as a deck of Magnetic Resonance Imaging (MRI) slices or Computed Tomography (CT) can be blended into a 2D X-ray image by firing rays through the volume and blending the voxels along the rays. This is a rather costly operation and the blending methods vary. The concept of volume rendering is also to extract the contours from given data slices. An iso-surface is a 3D constant intensity surface represented by triangle strips or higher order surface patches within a volume. For example, the voxels on the surface of bones in a deck of MRI slices appear to be the same intensity value.

4.3. Vector data – fluid visualization

From the study of turbulence or plasmas to the design of new wings or jet nozzles, flow visualization motivates much of the research effort in scientific visualization. Flow data are mostly 3D vectors or tensors of high dimensions. The main challenge of flow visualization is to find ways of visualizing multivariate data sets. Colors, arrows, particles, line convolutions, textures, surfaces, and volumes are used to represent different aspects of fluid flows (velocities, pressures, streamlines, streaklines, vortices, etc.)

4.4. Large datasets – computation and measurement

The visual presentation and examination of large data sets from physical and natural sciences often require the integration of terabyte or gigabyte distributed scientific databases with visualization. Genetic algorithms, radar range images, materials simulations, atmospheric and oceanographic measurements are among the areas that generate large multidimensional/multivariate data sets. The variety of data comes with different data geometries, sampling rates, and error characteristics. The display and interpretation of the data sets employ statistical analyses and other techniques in conjunction with visualization. The integration of multi-disciplinary data and information into visualization systems will help cross-disciplinary explorations and communications.

4.5. Abstract data – information visualization

The field of information visualization includes visualizing retrieved information from large document collections (e.g., digital libraries), the World Wide Web, and text databases. Information is completely abstract. We need to map the data into a physical space that will represent relationships contained in the information faithfully and efficiently. This could enable the observers to use their innate abilities to understand through spatial

relationships the correlations in the library. Finding a good spatial representation of the information at hand is one of the most challenging tasks in information visualization.

Many forms and choices exist for the visualization of 2D or 3D data sets, which are relatively easy to conceive and understand. For data sets that are more than 3D, visualization is still the most promising as well as challenging research area.

4.6. Interactive visualization

Interactive visualization allows us to visualize the results or presentations interactively in different perspectives (e.g., angles, magnitude, layers, levels of detail, etc.), and thus help us to understand the results on the fly better. GUI is usually needed for this interaction. Interactive visualization systems are most effective when the results of models or simulations have multiple or dynamic forms, layers, or levels of detail. Thus interactive visualization helps users interact with visual presentations of scientific data and understand the different aspects of the results. Interactive visualization makes the scientist an equal partner with the computer to manipulate and maneuver the 3D visual presentations of the abstract results.

4.7. Computational steering

Many software tools, packages, and interactive systems have been developed for scientific visualization. However, most of these systems are limited to post-processing of datasets. Any changes and modifications made to the model, input parameters, initial conditions, boundary conditions, or numerical computation considerations are accomplished off-line, and each change made to the model or input parameters causes most of the other steps and computation processes to repeat. In many cases the computations are done after hours of numerical calculations, only to find the results do not help solving the problems or the initial value of a parameter has been wrong. We would prefer to have a system in which all these computational components were linked so that, parameters, initial and boundary conditions, and all aspects of the modeling and simulation process could be steered, controlled, manipulated, or modified graphically through interactive visualization.

The integration of computation, visualization, and control into one tool is highly desirable, because it allows users to interactively "steer" the computation. At the beginning of the simulation before there is any result generated, a few important feedbacks will significantly help choose correct parameters and initial values. One can visualize some intermediate results and key factors to steer the simulation in the right direction. With computational steering, users are able to modify parameters in their systems as the computation progress, and avoid something being wrong or uninteresting after long hours of tedious computation. Therefore, computational steering is an important tool to adjust uncertain parameters, to steer the simulation in the right direction, and to fine tune the results. Implementation of a computational steering framework requires a successful integration of many aspects of visual supercomputing, including numerical computation, system analysis, and interactive visualization, all of which need to be effectively coordinated within an efficient computing environment.

4.8. Parallel coordinates

Parallel coordinates method represents a d-dimensional data as values on d coordinates parallel to the x-axis equally spaced along the y-axis (Figure 1, or the other way around rotating 90 degrees). Each d-dimensional data corresponds to the line segments between the parallel coordinates connecting the corresponding values. That is, each polygonal line of $p - 1$ segments in the parallel coordinates represents a point in d-dimensional space. Wegman was the pioneer in giving parallel coordinates meaningful statistical interpretations (Wegman, 1990). Parallel coordinates provides a means to visualize higher-order geometries in an easily recognizable 2D representation, which in addition helps find the patterns, trends, and correlations in the data set (Wegman and Luo, 1997).

The purpose of using parallel coordinates is to find certain features in the data set through visualization. Consider a series of points on a straight line in Cartesian coordinates: $y = mx + b$. If we display these points in a parallel coordinates, the points on a line in Cartesian coordinates become line segments in parallel coordinates. These line segments intersect at a point. This point in the parallel coordinates is called the dual of the line in the Cartesian coordinates. The point–line duality extends to conic sections. An ellipse in Cartesian coordinates maps into a hyperbola in parallel coordinates and vice versa. Rotations in Cartesian coordinates become translations in parallel coordinates and vice versa. More importantly, clustering is easily isolated and visualized in parallel coordinates. An individual parallel coordinate axis represents a 1D projection of the data set. Thus, separation between or among sets of data on one axis represents a view of the data of isolated clusters.

The brushing technique is to interactively separate a cluster of data by painting it with a unique color. The brushed color becomes an attribute of the cluster. Different clusters can be brushed with different colors and relations among clusters can then be visually detected. Heavily plotted areas can be blended with color mixes and transparencies. Animation of the colored clusters through time allows visualization of the data evolution history.

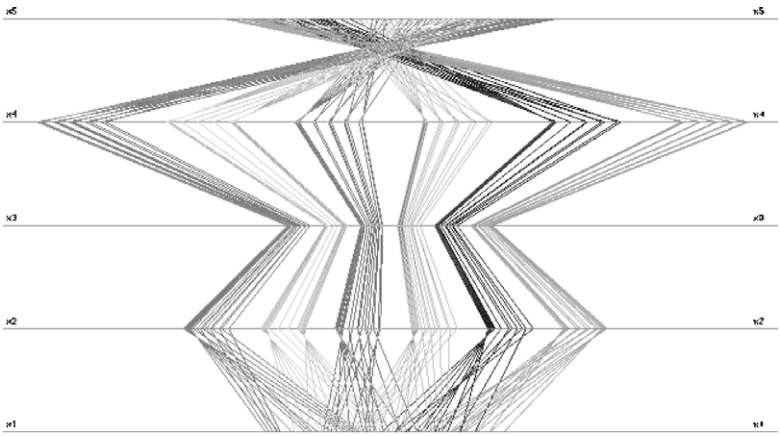

Fig. 1. Parallel coordinates (courtesy of E.J. Wegman and Q. Luo, 1997).

The tour method is to search for patterns by looking at the high dimensional data from all different angles (Buja and Asimov, 1985, Wegman and Luo, 1997). That is, to project the data into all possible d-planes through generalized rotations. The purpose of the grand tour animation is to look for unusual configurations of the data that may reflect some structure from a specific angle. The rotation, projection, and animation methods vary depending on specific assumptions and implementations. Currently, there are multidimensional data visualization tools that include parallel coordinates and grand tours available completely or partially for free, such as such as

ExplorN (ftp://www.galaxy.gmu.edu/pub/software/ExplorN_v1.tar),
CrystalVision (ftp://www.galaxy.gmu.edu/pub/software/CrystalVisionDemo.exe), and
XGobi (http://www.research.att.com/areas/stat/xgobi/).

The parallel coordinates method, which handles multiple dimensional data, is limited to data sets that have a few dimensions more than 3D. For data sets that have more or less hundreds of dimensions, parallel coordinates will be crowded and visualization will be difficult. Also, parallel coordinates treat every coordinate equally. There is no emphasis of certain scalar or vector values. It may be better to emphasis certain scalar values separately in different forms.

4.9. Linked micromap plots

Linked Micromap plots (LM plots) constitute a new template for the display of spatially indexed statistical summaries (Carr and Olsen, 1998). This template has four key features: displaying at least three parallel sequences of panels (micromap, label, and statistical summary) that are linked by position, sorting the units of study, partitioning the study units into panels to focus attention on a few units at a time, and linking the highlighted study units across corresponding panels of the sequences. Displaying LM plots on the Internet introduces a new and effective way of visualizing various statistical summaries. A reader can easily access and view the statistical data in LM plots everywhere around the World. A demo system with all the features is available on the website at http://graphics.gmu.edu/~jchen/cancer/. The following functions are implemented:

Display panels and study units. The LM plots for displaying the cancer summary statistics have four parallel sequences of display panels (columns), which are US/State micromap, State/County name, and two cancer statistical summaries. Figure 2 is a snapshot showing the basic layouts of the cancer statistical LM plots. In a row of a linked study unit, the geographic location, the leading dot before the name, and the statistics are all in the same color. On each statistics panel, the corresponding statistical value is shown as a dot with the confidence interval (CI) or bound displayed as line segments on both sides of the dot.

Sorting the study units. When sorting on a statistical variable, the displayed dot curve can show the relative relationships among the study units clearly. A sorting button with a triangle icon is located in front of the corresponding panel title. An up-triangle icon on the button represents an ascending sorting, and a down-triangle icon shows a descending sorting. Figure 2 shows that the study units (states) are sorted by one statistical summary (cancer mortality rate) in the descending order.

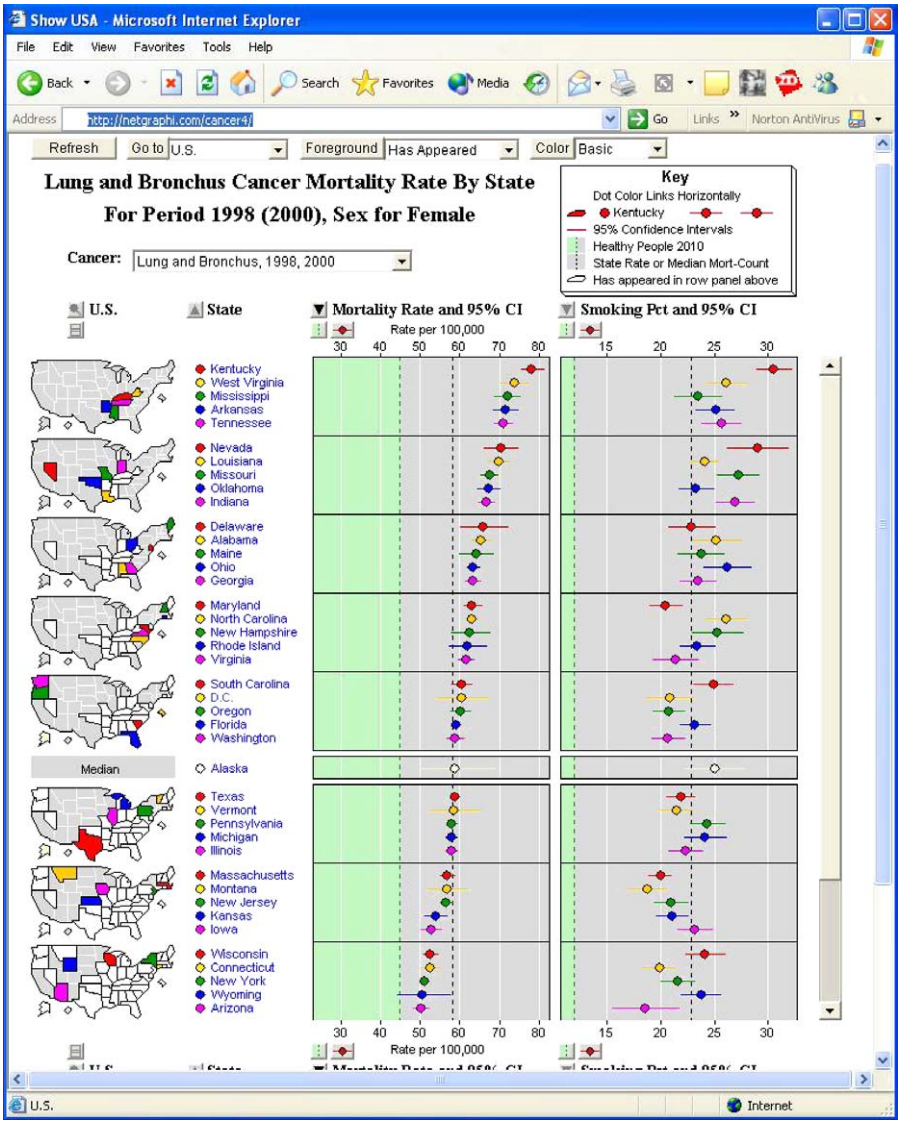

Fig. 2. A snapshot of the LM plots of the US cancer statistics. For a color reproduction of this figure see the color figures section, page 607.

Linking the related elements of a study unit. All the elements across the corresponding panels of the sequences are represented by the same color. When the user moves the mouse cursor onto any one of the related elements (micromap, unit name, or colored dots), all the linked elements in a study unit will blink. Meanwhile, the corresponding statistical summaries are displayed in text at the bottom of the browser window (in the browser's status line).

Grouping the study units. The study units are partitioned into a number of sub-panels vertically. This helps to focus attention on a few units at a time. In each group, a coloring scheme of five different colors is assigned to the five study units, respectively. This can show the linked elements of one study unit within a group more clearly.

Micromap magnification. Through the micromaps, a reader can find the locations of the study units intuitively. Some states may include many counties (study units). For example, Texas has over 200 counties. Since each micromap occupies a small fixed area, the area of a study unit in a micromap may be too small for a reader to see its shape. Magnifying the micromap helps the reader to view all study units clearly.

Drill-down and navigation. Drill-down is an operation that zooms in from a high-level LM plot to the corresponding low-level LM plot to view the detailed statistical data and the relationships among different hierarchical levels of detail. When using LM plots to visualize the national cancer statistical summaries, the US LM plot provides a starting place for subsequent drill-down to any other state LM plot. In the US LM plot, the state names and all the micromap regions are the active links to drill down to the corresponding state LM plots.

Overall look of the statistical summaries. When the number of the study units is bigger than the maximum that the display panel can hold, some study units will not be in the display panel. At this time, the reader cannot see the whole statistical curves formed by the dots shown in the statistical panels. Therefore, the overall look of the whole pattern presents very useful information. We have added a pop-up window to display the pattern in a scaled down fashion.

Displaying different statistical data sets. Interactively displaying different statistical data allows a reader to view and compare the relationships among different statistical results. In our system, we provide a pull-down menu for a reader to choose and view several different statistical cancer data sets. Once a reader selects a new cancer type, the data will be downloaded from the web-server and displayed in the display panel with a new calculated scale.

Statistical data retrieval. In a statistical visualization application, efficient retrieval of the statistical data is very important. In a statistical visualization system, the statistical data may be saved in different formats, places or modes to expedite retrieval. In our application, the statistical data is saved in the file mode and on the web-server. We do not need to write any code for the web-server. Instead, the Java applet directly retrieves the cancer data from the files.

The above set of web-based interactive LM plots provides a statistical data visualization system that integrates geographical data manipulation, visualization, interactive statistical graphics, and web-based Java technologies. The system is effective in presenting the complex and large-volume sample data on the national cancer statistics. With some modifications, the web-based interactive LM plots can be easily applied to visualize many other spatially indexed statistical datasets over the Internet.

4.10. Genetic algorithm data visualization

Genetic algorithms are search algorithms of better individuals based on the mechanics of natural selection and natural genetics (De Jong, 1990). The struggle for existence and better fitness that we see in nature is inherent in the algorithms. Some individuals have a higher probability of being recombined into a new generation and survive, while others have a higher probability of being eliminated from the population. To determine how good or bad a particular individual is, the genetic algorithm evaluates its *fitness* through the use of appropriate metrics on different fields. Each individual is an array of fields and a fitness value. Given an individual, its fitness is defined by the values of its fields. Many genetic algorithms produce data (individuals) of hundreds of fields with large population sizes over a long time (many generations). People are interested in visualizing the evolution of the population clusters and searching for the individuals that survive better, so to find good solutions to the target problem. The parallel coordinates method is limited to data of relatively ($p < 100$) low number of dimensions. We have to find some other methods to effectively visualize the very high-dimensional multivariate evolution data set. Also, the fitness is an important attribute in the visualization, which should be displayed in a separate scale. In the following, we introduce a dimension reduction method that transforms an arbitrary high-dimensional data into 2D with the third dimension representing the fitness.

Each individual encoding is a possible solution in a given problem space. This space, referred to as the search space, comprises all possible dimension field values and solutions to the problem at hand. We introduce a method called Quad-Tree Mapping (QTM) that uses the quad-tree subdivision method to map an ND data with a fitness value into a 3D scatter data at multi-levels of detail. Using animation and zooming techniques, the search space and the evolution process can be visualized. QTM is discussed in detail as follows.

Given an ND data, we use an m-bit binary value to represent each field value. At this point of time, we only consider the most significant bit in each field. Therefore, the N-dimension data is an N-bit binary. We take the first 2 bits in the data as the first level quad code, which has four possible values (00, 01, 10, and 11) placing the data into one of the four quadrants on a 2D rectangular plane (as shown in Figure 3). Then, we divide each quadrant into 4 sub-quadrants, and apply the next 2 bits in the data as the second level quad code placing the data into one of the four sub-quadrants on the 2D plane. For example, if the first two bits are 10, then the data will be in one of the upper-left sub-quadrants. This subdivision process continues recursively until all the bits are evaluated. If N is an odd number, we cut the last sub-quadrant into half instead of four pieces to make a decision. For each N-bit binary, there is a unique point on the 2D plane. Given

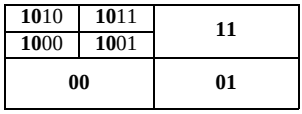

Fig. 3. QTM subdivision to map an N-dimensional data into a 2D space.

a point on a plane, we can calculate and find its corresponding N-bit binary data as well.

We can draw the 2D quad plane on the x–z plane and raise each data along the y-axis direction according to its fitness value. This way, we have a scatter plot of the whole gene population, search space, as well as the fitness values.

Because we only used one bit in the m-bit data value, there may be many overlapping points. This problem can be solved by the following methods. First, we can zoom-in to a specific point (area), which will invoke displaying the second bit of the original data set that have the same first N-bit array. Again, we can use the same method to display the second N-bit data. Therefore, we can zoom in and out m different levels corresponding to m bits of data. Second, after finishing the QTM subdivision for the first N-bit array, we can continue the subdivision with the second N-bit array. The process continues until we finish the mth N-bit array. Third, we can consider N m-bit data in a row. That is, we use the m-bit first field as an input to the QTM, and the second field, and so on. There are pros and cons for each of the methods. They can be interchanged or combined in a visualization tool. Of course, the plane can be divided into 6, 10 or other numbers of rectangles as well, instead of a fixed 4 (quadrants). Our CRT display usually has confined number of pixels (1280×1024). Therefore, if we subdivide an area so many times that the distance between the points are shorter than a pixel, the display will not able to do any better. Therefore, depending on the area used, the number of fields N, and the number of bits m of the given data set, we have to decide the specific criteria of subdivision or levels of detail.

After the QTM transformation, we have 3D scatter points of the individuals. At a low-level of detail where no overlapping points exit, we can use space curve fitting to find a fitness surface that smoothly goes through all the individual points. The surface represents the search space with all solutions. The peaks and ridges represent those

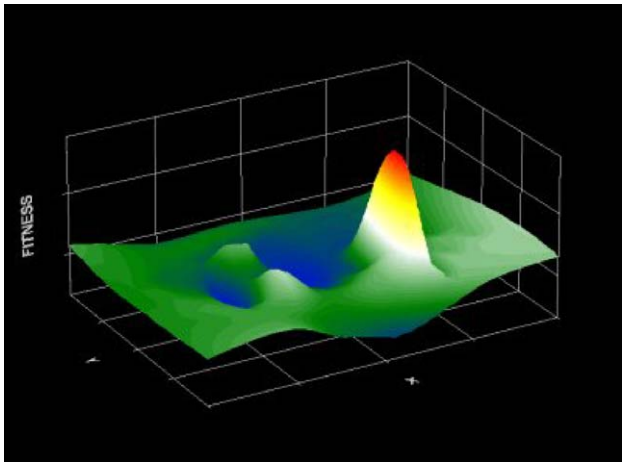

Fig. 4. An example of dimension reduction (QTM). For a color reproduction of this figure see the color figures section, page 608.

individual values (and possible solutions to the target problem) that are good. The bottoms and valleys represent the individuals that are likely to terminate. Figure 4 is an example of data with 226 fields, population size = 3, number of generation = 100, gene pool data is storied in an array of [300, 226].

5. Virtual reality

Virtual Reality (VR) extends 3D graphics world to include stereoscopic, acoustic, haptic, tactile, and other feedbacks to create a sense of immersion. A 3D image is like an ordinary picture we see, but a stereo image gives a strong sense of depth in 3D. It is generated by providing two slightly different views (images) of the same object to our two eyes separately. The head-mounted device (HMD), the ImmersaDesk/CAVE, and the VREX stereo projectors are different kinds of display devices for stereo images. A HMD has two separate display channels/screens to cover our two eyes. An ImmersaDesk or CAVE has only one channel like an ordinary display screen, except that it displays two different images alternatively for our two eyes. Viewers wear lightweight liquid crystal shutter glasses. These glasses activate each eye in succession. The glasses are kept synchronized with the two images through an infrared emitter. CAVE is the predecessor of ImmersaDesk, which is more expensive and has multiple display screens surrounding the viewers. An ImmersaDesk can be considered to be a one-wall CAVE. VREX's stereo projectors generate two images at the same time that can be viewed through lightweight, inexpensive polarized glasses. Wegman and his colleagues developed a mini-CAVE, which is a PC-based multiple stereo display system that saves both cost and space (Wegman, 2000, Wegman and Symanzik, 2002). Their papers (Wegman and Carr, 1993, Wegman, 2000, Wegman and Symanzik, 2002, Wegman and Luo, 2002) include valuable information concerning stereoscopic data visualization methods. They also include advantages and disadvantages of 1-wall versus 4-wall CAVES and an extensive history of stereoscopic visualization.

5.1. Hardware and software

The key hardware technologies in achieving VR are real-time graphics, stereo displays/views, tracking sensors, sound machines, and haptic devices. Real-time graphics (computer) and stereo displays (HMD, ImmersaDesk, CAVE, or VREX projectors) allow us to view stereoscopic scene and animation, and provide us a sense of immersion. Tracking sensors, which get the position and orientation of the viewer's head, hands, body parts, or other inputs, will enable us manipulate models and navigate in the virtual environment. Sound machines provide a sense of locations and orientations of certain objects and activities in the environment. Like sound machines, haptic devices vibrate and touch user's body, generating another feedback from the virtual environment in addition to stereoscopic view and 3D sound, enhancing the sense of immersion.

Some VR software tools are available that recognize well-defined commercial tracking sensors, sound machines, and haptic devices, in addition to functions in developing 3D virtual environment. Sense8's WorldToolKit and World_Up are cross-platform

software development system for building real-time integrated 3D applications. WorldToolKit also supports network-based distributed simulations, CAVE-like immersive display options, and many interface devices, such as HMDs, trackers, and navigation controllers. Lincom's VrTool is an OpenInventor-based toolkit to provide a rapid prototyping capability to enable VR users to quickly get their application running with the minimum amount of effort. MultiGen-Paradigm's Vega is a real-time visual and audio simulation software tool that includes stereo imaging. MR (Minimal Reality) Toolkit by the graphics group at University of Alberta is a set of software tools for the production of virtual reality systems and other forms of three-dimensional user interfaces.

5.2. Non-immersive systems

Often non-immersive 3D graphics systems are also called VR systems by some people. Users can change the viewpoint and navigate in the virtual world through input devices interactively. VRML (Virtual Reality Modeling Language) is a web-based 3D modeling and animation language – a subset of OpenInventor. Java3D, similar to VRML, is also a web-based graphics tool to assemble and manipulate predefined geometric models. DIVE (Distributed Interactive Virtual Environment) is an internet-based multi-user VR system where participants navigate in 3D space and see, meet and interact with other users and applications. Alice is a scripting and prototyping environment for 3D object behavior. By writing simple scripts, Alice users can control object appearance and behavior, and while the scripts are executing, objects respond to user input via mouse and keyboard.

5.3. Basic VR system properties

In an immersive VR system, users wear head-mounted devices (HMD) or special glasses to view stereoscopic images. The viewpoint usually follows the viewer's head movement in real time. In a non-immersive VR, which is usually a lot cheaper, users usually do not wear any device, and the viewpoint does not follow the user's head movement. Users navigate in the virtual world through input devices interactively and the image is usually a first-person view. In a VR system, navigation allows a user to move around and to view virtual objects and places, and interaction provides an active way for a user to control the appearance and behavior of objects. 3D navigation, probably with interaction, stereoscopes, and visualization, is the main property of a VR system, immersive or not.

Simulation is another property of a VR system. Simulations integrate scientific results and rules to control, display, and animate virtual objects, behaviors, and environments. Without simulation, the virtual world will not be able to describe and represent real world phenomena correctly. Different VR applications may simulate different objects, phenomena, behaviors, and environments, mostly in real time. These properties make the VR technology able to be applied in various areas such as data visualization, training, surgery, scientific studying, science learning, and game playing.

5.4. VR tools

A VR system often simulates certain real-world activities in various areas, such as training, education, and entertainment. A VR system always repeats the following processing steps:

(a) Handle user inputs from various devices – keyboard, mouse, VR trackers, sensors, voice recognition systems, etc.
(b) Calculate the new state of the objects and the environment according to the simulation models.
(c) Pre-process 3D objects including collision detection, levels of detail, clipping/culling, etc.
(d) Render the virtual world.

In order to achieve the above process, the software in the VR system has to be able to create a virtual world, handle various events from input devices, control the appearances and behaviors of the 3D objects, render the virtual world and display it on the display devices. In step (b), different VR applications may use different simulation models. No matter what application a VR system implements, the software to handle the other three steps, a high-level graphics library called a VR tool (or VR toolkit), is always needed. Therefore, VR tools, which are built on a low-level graphics library, are usually independent of the applications.

5.4.1. VR simulation tools

A VR system is usually a VR application implemented on top of a VR tool, which provides an API for the VR application to manipulate the objects according to the simulation models. VR tools are likely to be device dependent, built on low-level basic graphics libraries with interfaces to sensory devices. Some VR tools, such as MR Toolkit, OpenInventor, and WorldToolkit, only provide APIs embedded in certain programming languages for VR developers. It requires more knowledge and programming skills to employ these toolkits, but they provide more flexibility in application implementations. Others, such as Alice and WorldUp (often called VR simulation tools), provide graphical user interfaces (GUIs) for the developers to build applications. Developers achieve virtual worlds and simulations by typing, clicking, and dragging through GUIs. Sometimes simple script languages are used to construct simulation processes. VR simulation tools allow developing a VR system quicker and easier, but the application developed is an independent fixed module that cannot be modified or integrated in a user-developed program. A VR simulation tool, which is part of VR tools, is generally developed on top of another VR tool, so it is one level higher than the basic VR tools in software levels.

5.4.2. A list of VR tools

Some companies and academic departments have provided various VR tools for different computer platforms. Table 1 lists a few VR tools that are available and well-known. When we want to build a VR system, we do not have to develop everything by ourselves. Instead, we can use an existing VR tool and the necessary hardware, and add very limited application codes.

Table 1
A list of VR tools

Names	Platforms	Functions	References
ActiveWorlds	Windows	Network VR	http://www.activeworlds.com
Alice	Windows	VR simul. tool	http://www.alice.org
AVRIL	Windows	VR toolkit	http://sunee.uwaterloo.ca/~broehl/avril.html
DIVE	Windows	Network VR	http://www.sics.se/dive/dive.html
DIVERSE	Irix	VR toolkit	http://diverse.sourceforge.net
EON Studio	Multiple	VR toolkit	http://www.eonreality.com
GHOST	PC	VR toolkit	http://www.sensable.com
Java3D	Multiple	VR toolkit	http://www.java3d.org
MEME	Windows	Network VR	http://www.immersive.com
Summit3D	Windows	Web VR	http://www.summit3d.com
VREK	Windows	VR toolkit	http://www.themekit.com
Vega	Multiple	VR toolkit	http://www.multigen.com
VRSG	Multiple	VR toolkit	http://www.metavr.com
VrTool	SGI	VR toolkit	http://www.lincom-asg.com
X3D/VRML	Multiple	VR toolkit	http://www.web3d.org
WorldToolKit	Windows	VR toolkit	http://www.sense8.com/
WorldUp	Windows	VR simul. tool	http://www.sense8.com/

5.4.3. Basic functions in VR tool

In addition to a simulation loop and basic graphics functions, a VR tool usually provides the following functions as well:

Import that loads 3D objects or worlds from files on the hard disk into computer internal memory as data structures (called scene graphs) for manipulation and rendering. The 3D virtual world is usually generated with a 3D modeling tool.

Stereo display that allows two different projections of the VR environment to appear in our two eyes. For different display devices, such as HMD, CAVE, and Workbench, the display channels and operating mechanisms are different. A VR tool should support different display devices as well.

Event handling that accepts and processes user interaction and control. Various input from users and external devices are generated in the form of events. The event handling must be fast enough to guarantee the system to run in real time.

Audio and haptic output that generates sounds through the computer speaker or headphone and signals to drive the haptic devices.

Collision detection that prevents two objects to collide with each other and to touch or pick up virtual objects. Collision detection is a time-consuming operation, so most VR tools provide collision Enable/Disable switching functions for VR applications to turn it on/off if necessary.

Level of detail (LOD) that optimizes the rendering detail for faster display and animation. To provide LOD, a VR tool should save multiple different models for one object. VR tool will choose a corresponding model to render according to the distance between the viewpoint and the object.

User interface that accepts user inputs for data and status managements.

5.5. Characteristics of VR

We have briefly introduced VR. What does a high-end VR offer data visualization that conventional technologies do not? While a number of items could be cited, here is a list of those that are important:

Immersion, which implies realism, multi-sensory coverage, and freedom from distractions. Immersion is more an ultimate goal than a complete virtue due to the hardware limitations. For data visualization, immersion should provide a user with an increased ability to identify patterns, anomalies, and trends in data that is visualized.

Multisensory, which allows user input and system feedback to users in different sensory channels in addition to traditional hand (mouse/keyboard) input and visual (screen display) feedback. For data visualization, multisensory allows multimodal manipulation and perception of abstract information in data.

Presence, which is more subjective – a feel of being in the environment, probably with other realistic, sociable, and interactive objects and people. Presence can contribute to the "naturalness" of the environment in which a user works and the ease with which the user interacts with that environment. Clearly, the "quality" of the virtual reality – as measured by display fidelity, sensory richness, and real-time behavior – is critical to a sense of presence.

Navigation, which provides users to move around and investigate virtual objects and places not only by 3D traversal, but also through multisensory interactions and presence. Navigation motivates users to "visualize" and investigate data in multiple perspectives that goes beyond traditional 3D graphics.

Multi-modal displays, which "displays" the VR contents through auditory, haptic, vestibular, olfactory, and gustatory senses in addition to the visual sense. The mapping of information onto more than one sensory modality may well increase the "human bandwidth" for understanding complex, multivariate data. Lacking a theory of multi-sensory perception and processing of information, the critical issue is determining what data "best" maps onto what sensory input channel. Virtual reality offers the opportunity to explore this interesting frontier to find a means of enabling users to effectively work with more and more complex information.

6. Some examples of visualization using VR

Today, more complex data is generated and collected than ever before in human history. This avalanche creates opportunities and information as well as difficulties and challenges. Many people and places are dedicated to data acquisition, computing, and visualization, and there is a great need for sharing information and approaches. Here we list of some different research and applications in visualizing data using VR, so as to foster more insight among this fast-growing community. Visualization is a tool that many use to explore data in a large number of domains. Hundreds of publications address the application of virtual reality in visualization in the related conferences (IEEE VR, IEEE Visualization, ACM SIGGRAPH, etc.), journals (PRESENCE, IEEE CG&A, CiSE, etc.), and books (Burdea and Coiffet, 2003, Chen, 1999,

Göbel, 1996, Durlach and Mavor, 1995, Nielson et al., 1997, Rosenblum et al., 1994, Stanney, 2002, etc.). Below are specific projects that demonstrate the breath of applicability of virtual reality to visualization. The following examples are just demonstrative at random:

- archeology (Acevedo et al., 2001),
- architectural design (Leigh et al., 1996),
- battlespace simulation (Hix et al., 1999, Durbin et al., 1998),
- cosmology (Song and Norman, 1993),
- genome visualization (Kano et al., 2002),
- geosciences (Loftin et al., 1999),
- meteorology (Ziegler et al., 2001),
- materials simulations (Nakano et al., 1998, Sharma et al., 2002),
- oceanography (Gaither et al., 1997),
- protein structures (Akkiraju et al., 1996),
- software systems (Amari et al., 1993),
- scientific data (Hasse et al., 1997),
- statistical data (Arns et al., 1999, Wegman, 2000),
- vector fields (Kuester et al., 2001),
- vehicle design (Kuschfeldt et al., 1997),
- virtual wind tunnel (Bryson and Levit, 1992, Bryson, 1993, Bryson, 1994, Bryson et al., 1997, Severance et al., 2001).

References

Acevedo, D., Vote, E., Laidlaw, D.H., Joukowsky, M.S. (2001). Archaeological data visualization in VR: Analysis of Lamp Finds at the Great Temple of Petra, a Case Study. In: *Proceedings of the 2001 Conference on Virtual Reality, Archeology, and Cultural Heritage*. Glyfada, North Athens, Greece, November 28–30, pp. 493–496.

Akkiraju, N., Edelsbrunner, H., Fu, P., Qian, J. (1996). Viewing geometric protein structures from inside a CAVE. *IEEE Comput. Graph. Appl.* **16** (4), 58–61.

Amari, H., Nagumo, T., Okada, M., Hirose, M., Ishii, T. (1993). A virtual reality application for software visualization. In: *Proceedings of the 1993 Virtual Reality Annual International Symposium*. Seattle, WA, September 18–22, pp. 1–6.

Arns, L., Cook, D., Cruz-Neira, C. (1999). The benefits of statistical visualization in an immersive environment. In: *Proceedings of the 1999 IEEE Virtual Reality Conference*. Houston, TX, March 13–17, IEEE Press, Los Alamitos, CA, pp. 88–95.

Burdea, G.C., Coiffet, P. (2003). *Virtual Reality Technology*, second ed. Wiley.

Bryson, S., Levit, C. (1992). The virtual wind tunnel. *IEEE Comput. Graph. Appl.* **12** (4), 25–34.

Bryson, S. (1993). The Virtual Windtunnel: a high-performance virtual reality application. In: *Proceedings of the 1993 Virtual Reality Annual International Symposium*. Seattle, WA, September 18–22, pp. 20–26.

Bryson, S. (1994). Real-time exploratory scientific visualization and virtual reality. In: Rosenblum, L., Earnshaw, R.A., Encarnação, J., Hagen, H., Kaufman, A., Klimenko, S., Nielson, G., Post, F., Thalman, D. (Eds.), *Scientific Visualization: Advances and Challenges*. Academic Press, London, pp. 65–85.

Bryson, S., Johan, S., Schlecht, L. (1997). An extensible interactive visualization framework for the Virtual Windtunnel. In: *Proceedings of the 1997 Virtual Reality Annual International Symposium*. Albuquerque, NM, March 1–5, pp. 106–113.

Buja, A., Asimov, D. (1985). Grand tour methods: an outline. In: Allen, D. (Ed.), *Computer Science and Statistics: Proceedings of the Seventeenth Symposium on the Interface*. North-Holland, Amsterdam, pp. 63–67.

Carr, D.B., Olsen, A.R., et al. (1998). Linked micromap plots: named and described. *Statist. Comput. Statist. Graph. Newsletter* **9** (1), 24–31.

Chen, C. (1999). *Information Visualization and Virtual Environments*. Springer-Verlag, Berlin.

Chen, J.X., Yang, Y. (2000). 3D graphics formats and conversions. *IEEE Comput. Sci. Engrg.* **2** (5), 82–87.

Chen, J.X., Wang, S. (2001). Data visualization: parallel coordinates and dimension reduction. *IEEE Comput. Sci. Engrg.* **3** (5), 110–113.

Chen, J.X. (2002). *Guide to Graphics Software Tools*. Springer-Verlag.

De Jong, K. (1990). Genetic algorithm-based learning. In: Kodratoff, Y., Michalski, R.S. (Eds.), *Machine Learning: An Artificial Intelligence Approach, vol. 3*. Morgan Kaufmann, San Mateo, CA, pp. 611–638.

Durbin, J., Swan, J.E. II, Colbert, B., Crowe, J., King, R., King, T., Scannell, C., Wartell, Z., Welsh, T. (1998). Battlefield visualization on the responsive workbench. In: *Proceedings of the 1998 IEEE Visualization Conference*. Research Triangle Park, NC, October 18–23, IEEE Press, Los Alamitos, CA, pp. 463–466.

Durlach, N., Mavor, A. (Eds.) (1995). *Virtual Reality: Scientific and Technological Challenges*. National Academy Press, Washington, DC.

Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F. (1996). *Computer Graphics: Principles and Practice in C*, second ed. Addison–Wesley, Reading, MA.

Gaither, K., Moorhead, R., Nations, S., Fox, D. (1997). Visualizing ocean circulation models through virtual environments. *IEEE Comput. Graph. Appl.* **17** (1), 16–19.

Göbel, M. (Ed.) (1996). *Virtual Environments and Scientific Visualization 96*. Proceedings of the Eurographics Workshops in Monte Carlo, Monaco, February 19–20, 1996 and in Prague, Czech Republic, April 23–23, 1996, Springer-Verlag, Berlin.

Hasse, H., Dai, F., Strassner, J., Göbel, M. (1997). Immersive investigation of scientific data. In: Nielson, G.M., Hagen, H., Müller, H. (Eds.), *Scientific Visualization: Overviews, Methods, and Techniques*. IEEE Press, Los Alamitos, CA, pp. 35–58.

Hix, D., Swan, J.E. II, Gabbard, J.L., McGree, M., Durbin, J., King, T. (1999). User-centered design and evaluation of a real-time battlefield visualization virtual environment. In: *Proceedings of the 1999 IEEE Virtual Reality Conference*. Houston, TX March 13–17, IEEE Press, Los Alamitos, CA, pp. 96–103.

Kano, M., Tsutsumi, S., Nishimura, K. (2002). Visualization for genome function analysis using immersive projection technology. In: *Proceedings of the 2002 IEEE Virtual Reality Conference*. Orlando, FL, October 24–28, IEEE Press, Los Alamitos, CA, pp. 224–231.

Kuester, F., Bruckschen, R., Hamann, B., Joy, K.I. (2001). Visualization of particle traces in virtual environments. In: *Proceedings of the 2001 ACM Symposium on Virtual Reality Software and Technology*. November, ACM Press.

Kuschfeldt, S., Schultz, M., Ertl, T., Reuding, T., Holzner, M. (1997). The use of a virtual environment for FE analysis of vehicle crash worthiness. In: *Proceedings of the 1997 Virtual Reality Annual International Symposium*. Albuquerque, NM, March 1–5, p. 209.

Leigh, J., Johnson, A.E., Vasilakis, C.A., DeFanti, T.A. (1996). Multi-perspective collaborative design in persistent networked virtual environments. In: *Proceedings of the 1996 Virtual Reality Annual International Symposium*. Santa Clara, CA, March 30–April 3, pp. 253–260.

Loftin, R.B., Harding, C., Chen, D., Lin, C., Chuter, C., Acosta, M., Ugray, A., Gordon, P., Nesbitt, K. (1999). Advanced visualization techniques in the geosciences. In: *Proceedings of the Nineteenth Annual Research Conference of the Gulf Coast Section Society of Economic Paleontologists and Mineralogists Foundation*. Houston, Texas, December 5–8.

Nakano, O., Bachlechner, M.E., Campbell, T.J., Kalia, R.K., Omeltchenko, A., Tsuruta, K., Vashishta, P., Ogata, S., Ebbsjo, I., Madhukar, A. (1998). Atomistic simulation of nanostructured materials. *IEEE Comput. Sci. Engrg.* **5** (4), 68–78.

Nielson, G.M., Hagen, H., Müller, H. (1997). *Scientific Visualization: Overviews, Methods, and Techniques*. IEEE Press, Los Alamitos, CA.

Rosenblum, L., Earnshaw, R.A., Encarnação, J., Hagen, H., Kaufman, A., Klimenko, S., Nielson, G., Post, F., Thalman, D. (Eds.) (1994). *Scientific Visualization: Advances and Challenges*. Academic Press, London.

Severance, K., Brewster, P., Lazos, B., Keefe, D. (2001). Wind tunnel data fusion and immersive visualization. In: *Proceedings of the 2001 IEEE Visualization Conference*. October, IEEE Press, Los Alamitos, CA.

Sharma, A., Liu, X., Miller, P., Nakano, N., et al. (2002). Immersive and interactive exploration of billion-atom systems. In: *IEEE Virtual Reality Conference 2002*. Orlando, FL, IEEE Press, Los Alamitos, CA, pp. 217–225.

Song, D., Norman, M.L. (1993). Cosmic explorer: a virtual reality environment for exploring cosmic data. In: *Proceedings of the 1993 IEEE Symposium on Research Frontiers in Virtual Reality*. San Jose, CA, October 25–26, IEEE Press, Los Alamitos, CA, pp. 75–79.

Stanney, K.M. (Ed.) (2002). *Handbook of Virtual Environments: Design, Implementation, and Applications*. Lawrence Erlbaum Associates, Mahwah, NJ.

Vince, J. (1995). *Virtual Reality Systems*. Addison–Wesley, Reading, MA.

Vince, J. (1998). *Essential Virtual Reality Fast*. Springer-Verlag.

Wang, X., Chen, J.X., Carr, D.B., et al. (2002). Geographic statistics visualization: web-based linked micromap plots. *IEEE/AIP Comput. Sci. Engrg.* **4** (3), 90–94.

Watt, A.H., Watt, M. (1992). *Advanced Animation and Rendering Techniques: Theory and Practice*. Addison–Wesley, Reading, MA.

Wegman, E.J. (1990). Hyperdimensional data analysis using parallel coordinates. *J. Amer. Statist. Assoc.* **85**, 664–675.

Wegman, E.J., Carr, D.J. (1993). Statistical graphics and visualization. In: Rao, C.R. (Ed.), *Computational Statistics*. In: *Handbook of Statistics*, vol. 9. North-Holland, Amsterdam, pp. 857–958.

Wegman, E.J., Luo, Q. (1997). High dimensional clustering using parallel coordinates and the grand tour. *Comput. Sci. Statist.* **28**, 352–360.

Wegman, E.J. (2000). Affordable environments for 3D collaborative data visualization. *Comput. Sci. Engrg.* **2** (6), 68–72, 74.

Wegman, E.J., Symanzik, J. (2002). Immersive projection technology for visual data mining. *J. Comput. Graph. Statist.* **11** (1), 163–188.

Wegman, E.J., Luo, Q. (2002). On methods of computer graphics for visualizing densities. *J. Comput. Graph. Statist.* **11** (1), 137–162.

Ziegler, S., Moorhead, R.J., Croft, P.J., Lu, D. (2001). The MetVR case study: meteorological visualization in an immersive virtual environment. In: *Proceedings of the 2001 IEEE Visualization Conference*. October, IEEE Press, Los Alamitos, CA.