# Automatically Designed 3-D Environments for Intuitive Browsing and Discovery

Nelson L. Chang, Amir Said
Imaging Systems Laboratory
HP Laboratories Palo Alto
HPL-2003-92
April 28th , 2003*

E-mail: {nelson.chang, amir.said}@hp.com

automatically designed 3-D environments, virtual environments, intuitive browsing, discovery, human-computer interface, real time interactive visualization

While popular as a visual interface, current interactive 3-D environments are often times painstakingly created by hand, making it difficult to mass-produce custom environments quickly. This paper proposes interactive 3-D environments that are automatically designed on-the-fly based on various design rules, thereby separating layout and content. The resulting environments are compelling, customizable, and intuitive for information foraging. The interface also serves as an integrated framework for different types of rich media. The complete system for designing, constructing, and rendering the environments in real time is discussed and shown to have implications in a variety of applications including e-commerce.

* Internal Accession Date Only
Approved for External Publication

## 1. Introduction

The advantages of rich graphical user interfaces are by now well known. As technology advances, these interfaces will increasingly use three-dimensional (3-D) graphical or virtual environments that have been shown to be, for certain applications, extremely powerful, intuitive, and easy to use. However, despite their maturity, virtual environments still require the painstaking manual process of creation, achieved through a combination of programming and artistic skills. Even with latest design tools, most creations still require days, weeks, or even months of skilled artisanship for each new content.

A different approach is required to address the automatic creation of content in large scale. The Internet has shown the need to produce content by the thousands or millions. For instance, it would be impractical for web pages for e-commerce showing different products to be created individually one at a time. Hence, it is necessary to employ templates based on the database's metadata to automatically generate the required thousands of pages. Furthermore, one could introduce smart rules instead of simple templates that merely perform content substitution so that each page is unique and customized to the user's preferences and to its content.

This paper applies these ideas to the automatic creation of virtual environments. Instead of conforming to a rigid blueprint defined by constraints from the real world, one has an enormous amount of freedom when creating virtual environments as a means for communication. The proposed system creates not just a few environments from a slow and tedious manual process, but rather a seemingly infinite set of possible environments in real time.

An important note is that the proposed system does not sacrifice artistic creativity. The aesthetic component of the design can be integrated into the rules for environment creation. Thus, graphic artists and architects are still needed to create stylistic rules that are used in every synthetic environment, but in slightly different form (e.g. in a process similar to the current creation of templates).

The remainder of the paper is outlined as follows. Section 2 discusses keys to effective user interfaces. Section 3 highlights related work in the field of information visualization and virtual environments. Section 4 presents the proposed interface of virtual environment design automation and its main benefits. Sections 5 and 6 describe the details of the layout and presentation aspects of the interface, respectively. Section 7 shows experimental results of the proposed interface on various data collections. Finally, Section 8 concludes the paper and discusses future directions.

## 2. Effective User Interfaces

Devising effective means of visualizing large amounts of information is becoming increasingly important. This observation is especially evident on the Internet; the advent of the World Wide

Web brought with it an explosion of rich visual information available to any user with a simple click of the mouse.

In general, today's data collections are highly complex and visual in nature. They often consist of one or more attributes in the form of metadata with many different groupings and taxonomies. Some examples of related data collections include an online store's commercial inventory, a movie database, a collection of art pieces, or one's personal collection of digital photos and videos.

To manage the overwhelming wealth of information, a user requires a natural and compelling interface. There are a number of important issues for designing good interfaces. An interface should be intuitive and natural for most users. Users want to spend their time on exploring through the data and not worrying about how to operate the interface. Since users possess individual differences, the interface should be flexible enough to be tunable to each user and generate customized views.

Interaction is also an important part of a useful interface. It is crucial for the interface to provide low-latency control of data and good responsiveness. The user needs to be granted immediate and unobtrusive access to the information. The controls should give a lot of freedom but should not be overwhelming and cumbersome. Moreover, an interactive interface captivates and empowers the user with the seeming ability to manipulate.

Finally, the interface should be flexible enough to facilitate different information foraging modalities through databases [2]. For instance, a user may be interested one time in searching for a specific item and another time in browsing and discovering new items. An effective interface guides the user to find valuable information with minimum cost regardless of the particular task. It allows him/her to quickly gain insight, visualize relationships, form different connections easily, and make informed decisions. It also should account for the propensity of human beings to organize and classify information based on inherent connections to reduce complexity and simplify cognition.

It is informative to look at a shopping example to underscore the importance of good user interfaces. With the increased popularity of the Internet came the plethora of online stores to compete with traditional brick-and-mortar equivalents. These online storefronts typically consist of hypertext-based web pages as their primary user interface. The simple interface enables prospective buyers to shop in the comfort of their own home without having to drive to their local shopping mall and find parking. It also provides the convenience to shop from anywhere, any time of day, any day of the year. The online stores usually provide greater variety with lower cost since products do not actually consume valuable physical space. They are also particularly suited for focused searches, when the shopper knows and can enter appropriate search criteria or categories. More recently, online products often include "rotatable" image objects or even 3-D models to furnish the potential buyer with some form of visual interaction.

Despite claims to the contrary, however, these online stores never really took over and replaced brick-and-mortar stores, partly because they do not possess many of the key benefits

of physical stores. A typical brick-and-mortar store has been professionally designed to be pleasant and appealing to the senses. Natural spatial navigation facilitates easy recall of a product's location as compared to digging through nested links of web pages. In addition, the store's inventory is organized in an intuitive manner, allowing a shopper to effortlessly browse and discover new items. This important notion of discovery is still lacking in current online stores, despite attempts to address this inadequacy using ineffective software agents. One can easily imagine the difficulty in browsing for gifts online especially when one has only a vague concept of what to purchase.

## 3. Related Work

There has been a lot of work on designing effective interfaces in the area of information visualization. Many traditional interfaces emphasize the hierarchical nature of databases. Interfaces like the classical folder paradigm, hypertext-based browsing, and tree views [1] are easy to use and facilitate searching for known items. Other interfaces such as cone trees [10], hyperbolic trees [6], and pdq trees [5] visualize the explicit hierarchical structure, thereby giving context with the data. While it is difficult to compare these interfaces since their effectiveness depends highly on the required task, it is clear that these interfaces are not necessarily universally intuitive. Often, it is quite difficult to use these interfaces to discover new items or to remember previously found locations.

The above forms of navigation are typically aimed for text-based information and may be inadequate and inappropriate for visually rich data. A natural alternative is to use three-dimensional interfaces in the form of 3-D graphical or virtual environments. These types of interfaces mimic human beings' experiences in the physical world and create an intuitive spatial metaphor for information exploration.

3-D interfaces have become quite popular and common in recent years. Most children, and even many adults, already embrace 3-D and know these interfaces quite well primarily through games. The interfaces can be easily learned by the most fearful of people. This observation is especially true when the application, such as online shopping, does not involve an attacking monster and does not pressure the user for an immediate action or response. While they are more commonly found in the gaming and entertainment industries, 3-D interfaces are also beginning to populate the World Wide Web for applications like multiuser chat and collaborative shopping [3, 12, 13].

These kinds of 3-D interfaces are certainly not a new concept; they have been around in one form or another since the 1950s. Much of the early efforts focused on military applications and combat simulations. In later years, they were and continue to be successfully deployed in areas like distance learning, automobile manufacturing and design, and psychology to treat patients with acute fears. They have, however, become increasingly prevalent of late because today's computers and graphics hardware offer powerful graphics capabilities at relatively low cost. This fact means that more people have access to advanced machines capable of handling the demands of producing captivating environments. The interfaces' popularity will continue only to rise because of a constant surge in technology, led not surprisingly by the gaming and entertainment industries.

## 4. Virtual Environment Design Automation (VEDA)

Virtual environments enhance the user's experience through immersion. Generally, these environments are visually exquisite, due to the countless hours professional artists and designers spend on creating them. They facilitate navigation, interaction, and information access for visually rich data. However, the layouts of current environments are mostly immutable and are not dynamically constructed.

Devising effective layouts automatically is both crucial and difficult. It is important for the user to be able to access massive databases without feeling overwhelmed. There needs to be some sort of organization to make the data accessible. A simple approach would be to cluster the data into meaningful groups based on metadata. A further refinement would be to organize these groups according to some higher order organization or taxonomy. To explain one's unconscious inclination and expectation toward organization, it is natural to examine human psychology and the so-called Gestalt principles of perception [9]. One of the primary human perception cues is that of proximity or, in other words, objects that are placed closer together tend to be associated together. It should be clear that a more effective interface arises if these human tendencies are factored into the layout problem.

Automating layout design helps to separate the layout design from the actual content and to capitalize on the expertise of human designers, similar to the ideas found in VLSI design [11] and facility layout [8, 7]. Finding the optimal layout is usually NP-hard and needs heuristics to achieve an approximate solution in a reasonable time. Even so, such approaches may not be practical for an application that requires layout design in real time.

To address these issues, this paper proposes using visually interactive 3-D environments that are designed automatically. For a given database, the environment layout is not pre-canned but rather automatically constructed on-the-fly based on the associated metadata and various design rules (e.g. environment design rules, appearance rules, database-specific layout rules, and user preferences). These prespecified rules may range from simple conditions to more sophisticated ones from professional designers that dictate the layout process. In this way, one captures the essence of human design factors, something that may not be appropriate for automation.

As seen in Section 7, there are a number of benefits to the proposed virtual environment design automation or VEDA. It is a natural, effective user interface for browsing and discovery. For e-commerce applications, this improved interface captures many of the benefits of brick-and-mortar stores but in an online setting. Moreover, such an interface has implications beyond e-commerce, in areas such as education, collaboration, and personal multimedia management. Since it relates to human visual 3-D senses and memory, VEDA is more intuitive and easier to recall than current web pages. Because the layouts are rule based, it also provides a rich customizable experience that can easily and immediately accommodate user-specified redesign as well as dynamically changing data. Furthermore, it provides an integrated visual framework for various rich media (e.g. text, audio, images, video, 3-D models, "rotatable" 3-D objects, etc) to allow for greater visual interaction and generate an

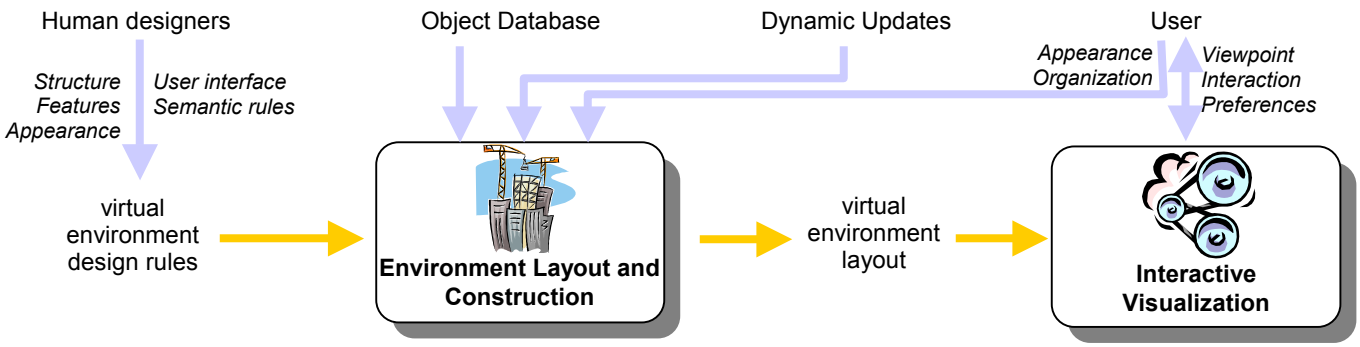



**Figure 1. Block diagram of the proposed VEDA system.**

even more compelling experience. The integrated framework also presents the rich media in a proper context to keep the virtual metaphor consistent for the user.

The entire VEDA system, diagrammed in Figure 1, consists of two primary components described in the upcoming sections. The first component designs and builds the given 3-D environment given various design rules and object information. The output is a description of the environment layout. The second component uses this description to provide a rich interactive visualization of the environment for the user. The user interacts with this component to specify changes in the presentation of the data (e.g. changing virtual viewpoint, activating linked rich media, specifying instant redesign).

## 5. Layout and Construction

The first stage of VEDA focuses on generating the layout of a given environment by using various prespecified design rules along with the object database and user specifications. Ultimately, these design rules will come from professional designers, such as artists and interior designers, who specify characteristics of the environment to make it aesthetically pleasing and appeal to human perception cues.

There are three main categories of design rules. The first consists of environment design rules that detail the structure and main features of permissible environments. Currently, VEDA uses environment design rules that specify an environment comprised of adjacent rectangular partitioned areas (e.g rooms). Moreover, these rooms are organized such that nearby rooms are somewhat related in organization. Each room is constrained to be spacious enough to accommodate all its associated images and reduce the occurrence of very long or skinny rooms.

Another category includes appearance rules that dictate the style and type of architecture of the environment. These rules help to make the constructed environments appealing to a broad class of users. Some examples include gothic-style architecture, ornate columns, and different textures for the ceilings, walls, and floors.

Finally, database-specific layout rules specify the available organizations for a particular database. These rules are functions based on the object attributes and metadata. The

current VEDA system employs XML-based rules such as ordering a movie database alphabetically by genre or clustering a commercial database by product family hierarchy.

Along with the design rules, the object database itself plays an important role in the VEDA layout. As mentioned, the object database consists of a collection of objects, where each object has an appropriate visual representation (i.e. image) and a number of attributes in the form of associated metadata. The attributes define various characteristics of a given object and not all attributes have to be specified for every object. For instance, all objects of a movie database have attributes such as movie title, director, and actors, whereas only some of the movies have attributes such as MPAA rating, historical music advisor, or aquatic researcher. VEDA presently uses XML-based databases, however it could just as easily work with typical relational databases. In addition, each object may have additional attributes that link to one or more related rich media such as URLs, audio clips, images, video clips, and 3-D models.

The complete layout process begins with the user selecting from the predetermined design rules and preferences for a particular database. The system clusters the database with respect to the user's choice and the database's metadata. The next couple of subsections focus on a particular algorithm that VEDA performs for automatic construction.

### 5.1 Environment Layout

The first aspect of automatic layout carves the environment into partitioned areas (i.e. rooms) and assigns the appropriate subset of the database to each area. A proposed layout scheme uses the selected design rules to generate the associated layout tree. The virtual environment is constructed through recursive partitioning of the environment space according to the layout tree. For convenience, the partitioned areas are defined to be adjacent rectangular rooms with zero width partitions.

The layout tree is constructed to encapsulate the chosen design rules applied to the object database. In particular, the leaf nodes of the layout tree point directly to the appropriate objects in the database. For example, if a design rule consists of alphabetically sorting a movie database by genre, VEDA forms subtree clusters based on the movie's genre. These clusters are then arranged alphabetically as sibling nodes and the root node is set to point to these cluster nodes. An alternative design rule could consist of following a taxonomy, such as the hierarchical organization of products. In this case, the layout tree resembles the taxonomy tree, where the leaf nodes point directly to pertinent products. Note that it is possible for multiple leaf nodes to point to the same object; this common condition occurs when the object belongs to two or more clusters simultaneously based on the design rules.

Once the layout tree has been established, VEDA uses a top-down environment construction algorithm to partition the environment into meaningful and spatially related rooms. The algorithm is similar to Johnson and Shneiderman's tree-map algorithm [4] when regarding the environment as a 2-D floor plan. The proposed algorithm starts with a given rectangular volume to specify the overall dimensions of the environment. For each level of the layout tree, the algorithm divides the children nodes of the current parent node into approximately equal

subtrees based on total number of leaf nodes. For faster processing, the algorithm assumes a left-to-right ordering in the layout tree and does not consider rearranging the tree.

After determining the split point for the current set of nodes, VEDA performs binary space partitioning on the environment with guillotine cuts. In other words, it determines whether to use a vertical partition or a horizontal partition to subdivide the current bounding box. For a $w$x$h$ bounding box, a vertical partition separates the width of the bounding box into $x$ and $w$-$x$ such that $x = wn/N$, where $n$ is the number of leaf nodes in the subtree and $N$ is the total number of leaf nodes in the parent's subtree. The ratio $n/N$ effectively compares the current node's leaf nodes to its siblings' leaf nodes. Likewise, a horizontal partition separates the height of the bounding box into $y$ and $h$-$y$ such that $y=hn/N$. Then, the partition that maximizes "squareness" of the two newly formed rooms is selected, thus reducing the appearance of "skinny" rooms (i.e. one dimension is much longer than the other). Specifically, the horizontal partition is chosen if $|x-h|+|w-x-h| < |y-w|+|h-y-w|$; otherwise, the vertical partition is selected. Instead of using absolute differences, one could minimize aspect ratios of the candidate bounding boxes to determine which partition to select.

The algorithm continues recursively down the layout tree until no more branches can be processed. When a particular subtree has no more branches, the corresponding room becomes fixed in dimension and the associated leaf nodes are assigned to these rooms. Portals are automatically constructed between adjacent rooms by identifying shared partitions and subdividing the walls if the shared partition length is larger than some prescribed portal dimension. In this way, a room consists of four or more walls.

Figure 2 shows an example to illustrate this algorithm. In each subimage, the layout tree is shown on the left side while the overhead view of the current environment space is diagrammed on the right. Starting with given dimensions of the environment, one traverses top-down to the first level of nodes in the layout tree. VEDA divides the tree to balance the number of leaf nodes in each division. In this case, the division occurs between the third and fourth parent node resulting in a total number of six and seven leaf nodes in each division, respectively. Both vertical and horizontal partitions are examined and a vertical partition, drawn as a dark line segment, is found to maximize "squareness" for the subdivided bounding boxes. Notice that the area of the bounding boxes is proportional to the number of leaf nodes in the subtree; the light gray box consists of 6/13 of the total environment area corresponding to the six out of 13 leaf nodes in the total layout tree. VEDA proceeds to traverse down the remaining levels in the layout tree and performs similar binary grouping and partitioning on every subsequent level of nodes. Note that the numbers in the diagram represent the number of leaf nodes assigned to the room. As seen in the final subimage, portals between adjacent rooms are calculated once the room dimensions have been finalized.

In the end, the approach produces an environment subdivided into adjacent rectangular rooms, each with a subcollection of objects associated with it. The proposed approach has numerous benefits. The top-down layout algorithm determines the number and dimensions of the walls defining every room, and it is computationally linear in the number of nodes in the layout tree. It ensures that the floor area of each room is proportional to the number of objects corresponding to that room. Hence, at a glance, one can compare the number of objects

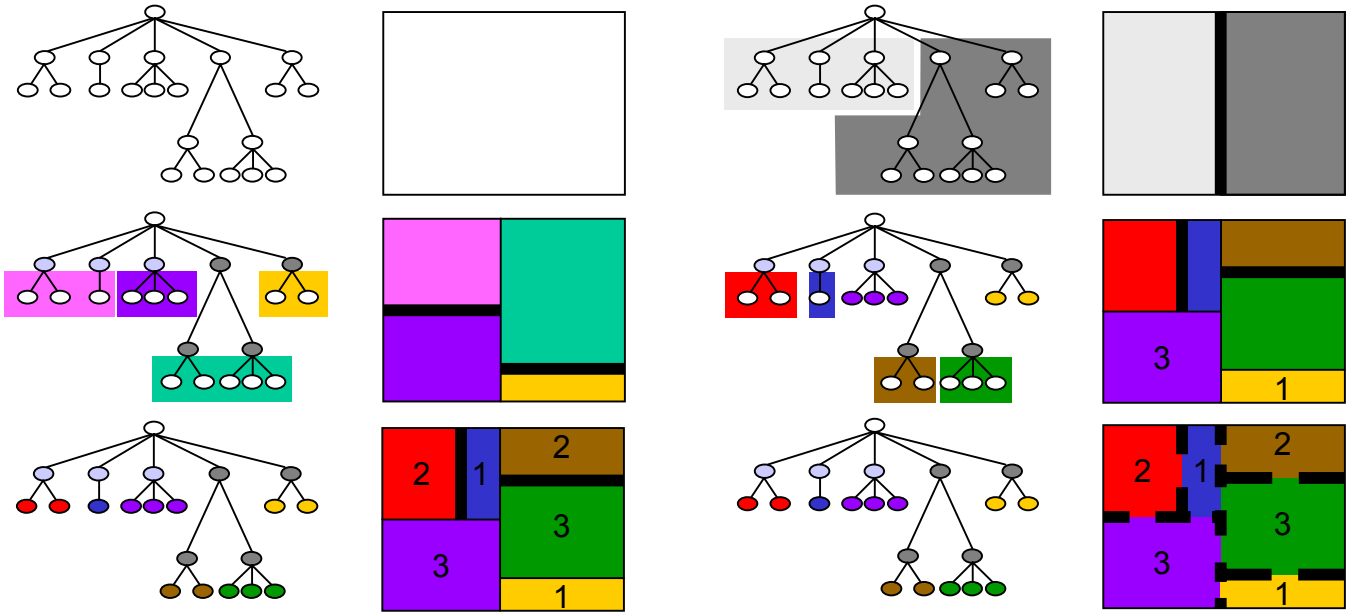



**Figure 2. Evolution of recursive layout algorithm to derive rooms given layout tree.**

across rooms. The resulting layouts enforce spatial proximity of rooms corresponding to nodes at the same level in the layout tree. This behavior follows the Gestalt proximity principle that items that are spatially near one another are likely be related and vice versa. In addition, this approach is flexible enough to enable multiple rules to be applied to different portions of the layout tree, thereby providing customized organization of the data.

### 5.2 Appearance Design

The next step focuses on distributing the assigned objects to the walls of each room and presenting the constructed rooms in an aesthetically pleasing manner. It determines location and placement of the visual representations of the objects (herein referred to as image objects) in every room as well. It also defines the appearance style for the environment.

To start, the layout algorithm distributes the image objects assigned to every room so as to maximize their dimensions and balance their distribution within the room while preserving their aspect ratios. For every room, the algorithm uses a weighted function of the image objects' aspect ratios and the dimensions of the walls to determine the subset of image objects should be placed on which walls. For computational speed up, one assumes a left-to-right ordering of the image objects and clockwise wall ordering starting with the north wall. In this case, the image objects are split into successive subsequences by minimizing the difference of cumulative sum of aspect ratios and cumulative sum of wall widths.

Once the image objects have been assigned to the individual walls, the algorithm then automatically selects the appropriate number of rows of image objects to maximize the overall size of the images constrained within the wall's dimensions. This helps to avoid creating rooms with a large number of extremely small image objects lying along a single row. Suppose *r* is the number of image objects rows for a wall with *w*x*h* dimensions. Then, the wall

may be considered having *r* horizontal subwalls, each with dimension *wx(h/r)*. The number of rows *r* is selected that maximizes the sum of the scaled image objects that still fit inside each of the subwalls. For uniformity, one can choose *r* to be constant for all walls in the same room by summing over all walls simultaneously.

The image objects are finally placed at specific locations on each subwall based on the chosen style. By default, the image objects are centered both horizontally and vertically in their subwall. Custom interobject spacing and object alignment may be specified and factored in during the distribution phase.

Additional style considerations are provided to enhance the appearance of the environment. A decorative frame and simulated individual spot lighting may be introduced. Every wall is given a true 3-D shape to provide realism. The walls of each room are pseudo-randomly colored to make each room distinctive to the user. These colors may be altered based on the user's actions and the organization of the database. Tiled image textures are used for the floor, walls, and ceiling to improve overall aesthetics. Different combinations of these parameters may be formed to create different "atmospheres," such as an indoor hotel environment with marble floors or an outdoor rock garden.

## 6. Interactive Visualization

Once the environment has been designed and constructed, it can be rendered and the user can interact with it. The following subsections detail both of these issues.

### 6.1 Real-time Rendering

The constructed environment consists of mostly images (i.e. image objects, texture maps) with limited geometry (i.e. wall partitions). Visualizing such an environment demands fast and efficient rendering algorithms. The rendering algorithms of many games may not be applicable since they tend to have distinct worlds with highly reusable textures and more significant geometry. Because of limited texture memory, VEDA requires a smart mechanism to decide what image textures should be displayed.

The proposed rendering algorithm for VEDA performs many speed enhancements to ensure real-time rates. At every time instant (typically every 1/30 second), it recursively traverses the hierarchical list of graphical objects representing the overall environment based on the user's current position. The algorithm performs portal culling (i.e. recursive visibility check of adjacent rooms through the portals) to substantially reduce the number of graphical objects in the list to the candidates that are actually visible to the user. As shown in Figure 3, portal culling begins in the user's current room, indicated by the dot in the lower left corner, and propagates outward to the nearest adjacent rooms via the visible portals. In these calculations, floors and ceilings are also considered portals to handle more generalized environments and viewing conditions. The current user's viewing frustum is intersected with every graphical object in the list to determine the graphical objects that are currently visible. Note that only the portions of the rooms that are actually visible are rendered.

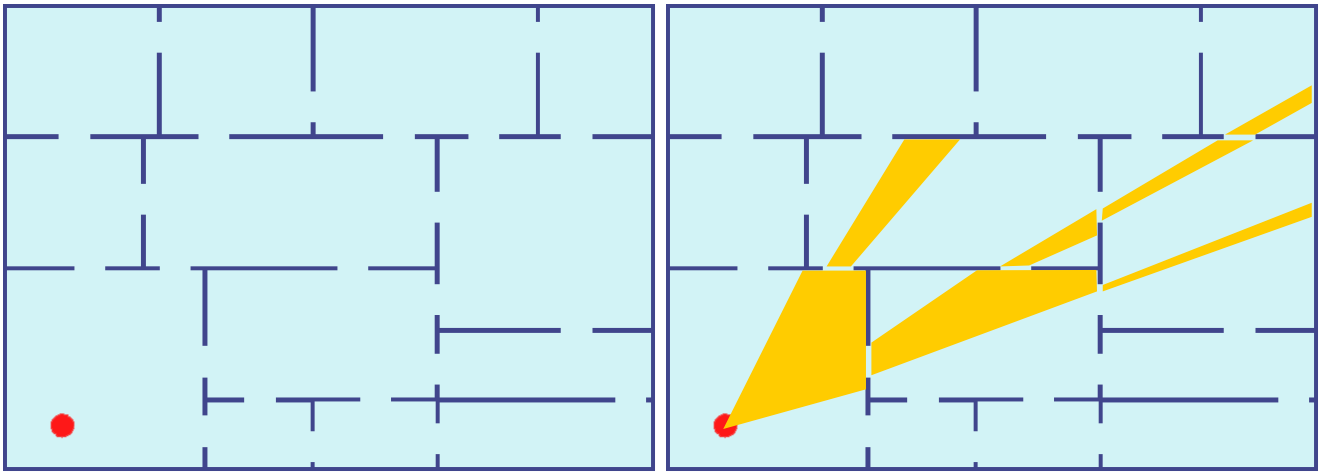


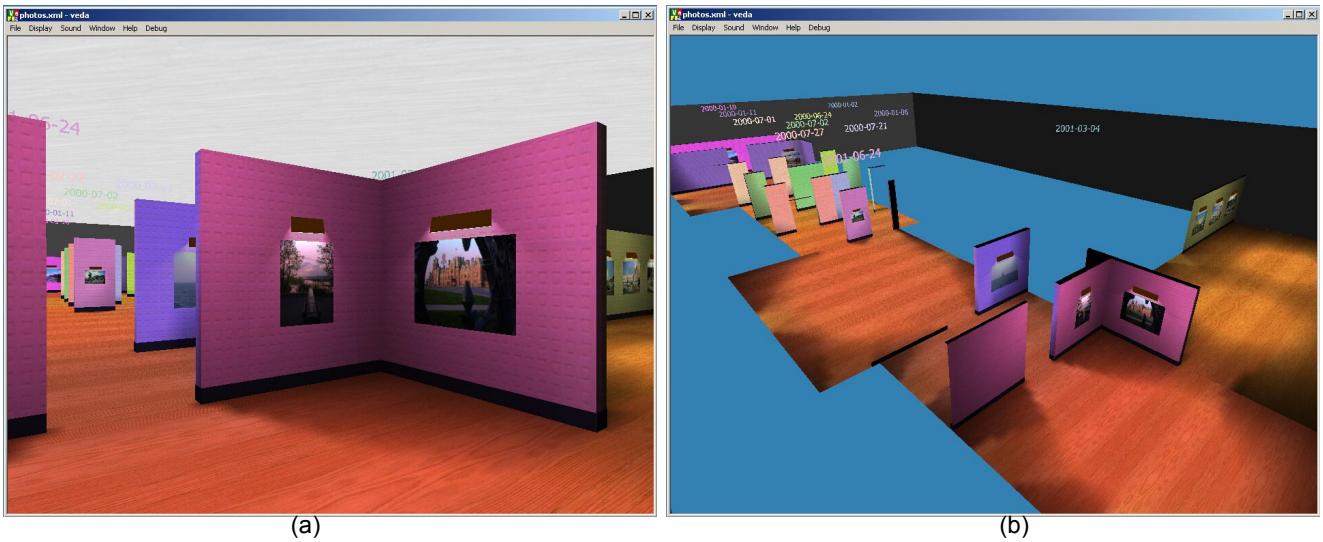
**Figure 3. Example of portal culling to reduce the rendering load in VEDA.**

The image objects situated on included graphical objects are then considered. The bounding box of each of these image objects is projected to the user's viewpoint and its projected area is used to assess the appropriate level of detail and resolution level. Distant image objects are adaptively rendered at very low resolutions while close up image objects require much higher resolution. An iterative algorithm estimates the time to transfer the compressed images from main memory or disk to texture memory. If there is extra time (e.g. the user is sitting idle), the image objects automatically increase resolution. VEDA thus maintains interactive rates while presenting the user with the highest available image resolution.

An example of applying the rendering algorithm is found in Figure 4. Figure 4 (a) shows a typical screenshot of the environments designed by VEDA. To enable interactive visualization, one renders only the visible graphical objects and culls the rest. Figure 4 (b) shows the graphical objects of the environment that are actually required to render the view in Figure 4 (a). It is important to note that only the visible image objects and partitions are included, which considerably reduces the rendering load. One could increase granularity of the graphical objects so that, for instance, only the exact portion of the floor or the specific letters of the signage are rendered. However, these improvements do not drastically improve the rendering rate for the tested databases.

Additional rendering elements, including efficient lighting and shadow effects, further enhancing the user's overall experience. The light sources are assumed to be centrally located in the room, where the number and location vary based on the overall dimensions of the room. Lighting maps and shadows are precomputed to a first order approximation for adjacent rooms. This computation needs to be performed only once for a given environment layout. The resulting lighting maps are modulated with the various textures (e.g. walls, floors) to efficiently simulate lighting and shadow effects. This solution gives a reasonable appearance with minimal computation as compared to performing more intensive lighting calculations on-the-fly.

### 6.2 Intuitive User Interface




(a) (b)

**Figure 4. Rendering examples of VEDA: (a) showing the view from the user's current perspective, and (b) showing only the rendered graphical objects for this perspective.**

In addition to rendering issues, it is important to make the 3-D environment user friendly and provide a lot of useful cues. Numerous user interface elements are added to improve usability. The user can seamlessly navigate around and fly throughout the virtual environment using a keyboard, mouse, or joystick. The system reduces the number of degrees of freedom for movement by eliminating rolling motion and by tying tilting motion with changes in height. This reduction helps to lessen the confusion with navigating in free space. The system also enables automatic transition between navigation mode and an overhead, bird's eye viewing mode.

With respect to graphics enhancements, the user can display his/her path in the environment. Various 3-D signage is provided throughout the environment to emulate signs available at many department stores. The signage automatically rotates with the user's viewpoint to improve visibility. Optional dialog windows furnish information about the metadata as well as the user's position with respect to an overhead map of the environment. The ceilings of each room are rendered in a translucent mutable color to enable visibility into every room and to give cues about how the rooms are implicitly grouped.

There are also numerous modes of interaction throughout VEDA. A brief description of any image object is instantly obtained simply by hovering the mouse cursor over the desired object. The user can click on any image object of interest and be automatically moved toward that object using Dijkstras shortest path algorithm and linear interpolation of viewpoints. Alternatively, the user can click and activate the associated rich media links to an image object, thereby starting the integrated playback and simulating interaction. For example, the user can launch a corresponding web page, start playing a related audio or video clip, view a "rotatable" 3-D object as a digital hologram embedded in the environment, toggle the visibility of 3-D models, and so forth.

Additionally, the interface also allows the user to immediately redesign the environment based on a different set of rules. Similarly, the user can change the style or physical appearance of the environment. Furthermore, the user can also search through the database by typing in a

few keywords. A virtual search room is automatically created that hovers above the user's current position featuring the search results. The user then clicks on any of the returned image objects and will be automatically flown to the appropriate position in the environment.

## 7. Results

The current system consists of customized layout and visualization software written in C++ for the Windows platform using OpenGL and DirectX 8.1. VEDA needs only a moderately fast computer with a reasonably powerful graphics card to generate compelling layouts in only a few seconds. The results shown in this section come from a 800 MHz Pentium III with an Elsa Gloria II graphics card with 64 MB of texture memory.

The object databases with associated metadata, as well as the layout rules, are described in XML, and three example databases are presented in this section. VEDA has been tested with many different databases, including a 4000+ high-resolution image database and even a collection of music tracks for audio browsing. All of the databases run at interactive rates, giving the user a responsive and visually rich interface.

While the results are much more impressive when directly interacting with the system, screenshots give an adequate representation. Figure 5 shows screenshots for a database consisting of 250 movies and their associated metadata. After the user selects to visualize by genre, the system automatically constructs the layout shown in Figure 5 (a). Many of the user interface elements described above are clearly apparent, including naturally partitioned areas, compelling textures and lighting effects, optional metadata and map dialog windows, multiple image object row layouts, and automatically rotating signage. Figure 5 (b) demonstrates how rich media, such as the 3-D model chair and the movie trailer playing in the background, can be activated and integrated directly to the environment, creating a consistent and immersive atmosphere. Figure 5 (c) presents the view as the user defies gravity and "flies" above the environment. Figure 5 (d) shows the layout automatically redesigned after the user selects to visualize by director.

Figure 6 shows a database consisting of a subset of HP products laid out according to the product taxonomy. This example emphasizes the hierarchical nature of the layout algorithm. For example, the printing products are automatically organized in rooms on the left side of Figure 6 (a), and these products are further classified into subcategories (color, black and white, photo, multifunction, and large format) in Figure 6 (b). Similar organization occurs for the digital imaging and computing products. Notice that the layout algorithm naturally places rooms of similar content spatially near one another to form pseudo-"departments." Figure 6 (c) shows a "rotatable" 3-D camera product directly integrated into the virtual environment that updates its view based on the user's viewpoint to form a type of digital hologram. Figure 6 (d) shows an example of a virtual search room that is formed after the user enters the desired keywords and links to image objects distributed throughout the environment.

VEDA may also be used for non-commercial applications. Figure 7 is an example for a collection of digital photos. In this case, the metadata and design rules come directly from EXIF data found in digital camera JPEG images. Thus, a user's personal photo collection can

be captured and instantly visualized in this environment without additional work and without requiring special rules. Figure 7 (a) organizes the photo collection based on the date, while Figure 7 (b) is based on subject distance from the camera. In the latter case, the up-close macro shots and distant landscape shots are automatically separated and clustered appropriately without any additional work. One can further postprocess the images to augment the metadata. As shown in Figures 7 (c) and (d), respectively, owner information and even geospatial location information could be easily introduced to the images and viewed in this framework. In the future, faces in the photos could be recognized and added to the metadata. It should be evident from these images that the resulting environments present one's digital photos in a type of compelling virtual museum. The screenshots also highlight the flexibility of the system to allow immediate changes in appearance (e.g. indoor environment, museum environment, even outdoors).

## 8. Conclusions

VEDA is a powerful 3-D interface for information exploration and interaction. Data collections that are highly visual in nature, as opposed to text documents or file systems, would benefit the most from this interface. Its main contributions consist of the following:

- Unlike traditional 3-D interfaces, the layouts are automatically organized and designed in real time based on various prespecified rules and user preferences, promoting the notion of information discovery.
- VEDA is a pleasant, intuitive, responsive, and customizable environment that gives users a lot of freedom without feeling overwhelmed.
- It uses the available screen space more effectively as compared to traditional hierarchical interfaces. It also displays focus and context without having to stress the hierarchical connections of the layout tree.
- It employs efficient rendering algorithms for real-time interaction.
- It provides a natural framework to seamlessly integrate rich media together into a single coherent environment.

While it has implications in many applications, VEDA is particularly useful for e-commerce and may be easily deployed in numerous ways. It could serve as the future of online shopping and web storefronts with a 3-D interface that is much more intuitive and useful than current web-based html pages. It could also be deployed as helpful physical kiosks found within department stores. In this case, VEDA marries the benefits of traditional brick-and-mortar stores with the convenience of their online counterparts. Finally, it could be distributed as a compelling mass-mailed CD catalog consisting of dynamically updated inventory and prices.

The presented work serves as an important first step toward the vision of automatic creation of virtual environments and there are many exciting directions one can take. A more thorough study of design rules will lead to more aesthetically pleasing virtual environments. It is also important to consider semi-automated tools to simplify layout and environment design. Scaling the system to accommodate very large databases requires further examination.

Efficient representation and distribution of the environments will allow for multi-user collaboration and seamless rich media communication.

## 9. References


[1] K. Andrews, "Visualizing Cyberspace: Information Visualization in the Harmony Internet Browser," InfoVis'95, pp. 97--104.

[2] C. Chen, *Information Visualization and Virtual Environments*, Springer-Verlag, London, 1999.

[3] R. Hawkes and M. J. Wray, "LivingSpace: A Living Worlds Implementation using an Event-based Architecture," *HP Labs Tech. Rep.*, HPL-98-181, 1998.

[4] B. Johnson and B. Shneiderman, "Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures," *Proc. IEEE Vis, Conf.'91*, pp. 284--291.

[5] H. P. Kumar, C. Plaisant, and B. Shneiderman, "Browsing Hierarchical Data with Multi-level Dynamic Queries and Pruning," *Int'l J. Human-Comp. Studies*, 46(1), pp. 103—124, 1997.

[6] J. Lamping and R. Rao, "The Hyberbolic Browser: A Focus+Context Technique for Visualizing Large Hierarchies," *J. Vis. Lang. Comp.*, 7(1), pp. 33--55, 1996.

[7] R. S. Liggett, "Automated Facilities Layout: Past, Present, and Future," *J. Auto. Construction*, 9(2), pp. 197--215, March 2000.

[8] R. D. Meller and K. Y. Gau, "The Facility Layout Problem: Recent Trends and Perspectives," *J. Manu. Sys.*, 15(5), pp. 351—366, 1996.

[9] K. Mullet and D. Sano, *Designing Visual Interfaces: Communication Oriented Techniques*, Englewood Cliffs, NJ, Prentice Hall, 1995.

[10] G. G. Robertson, S. K. Card, and J. D. Mackinlay, "Information Visualization using 3-D Interactive Animation," *Comm. ACM*, 36(4), pp. 57--71, 1993.

[11] S. M. Sait and H. Youssef, *VLSI Physical Design Automation: Theory and Practice*, McGraw-Hill, NY, 1995.

[12] "Virtual Worlds for E-Commerce," *Blaxxun Interactive White Paper (www.blaxxun.com )*, 2001.

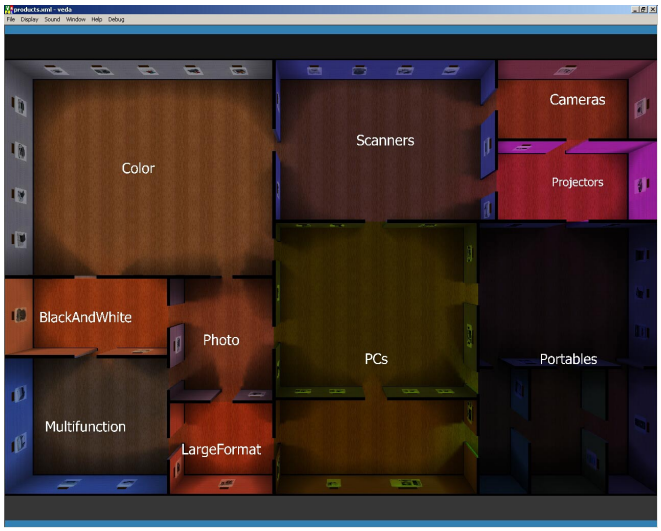
[13] "Welcome to the Home of the 3D Internet, Virtual Reality, and Community Chat," *Active Worlds web site (www.activeworlds.com)*, November 2002.


(a) (b)

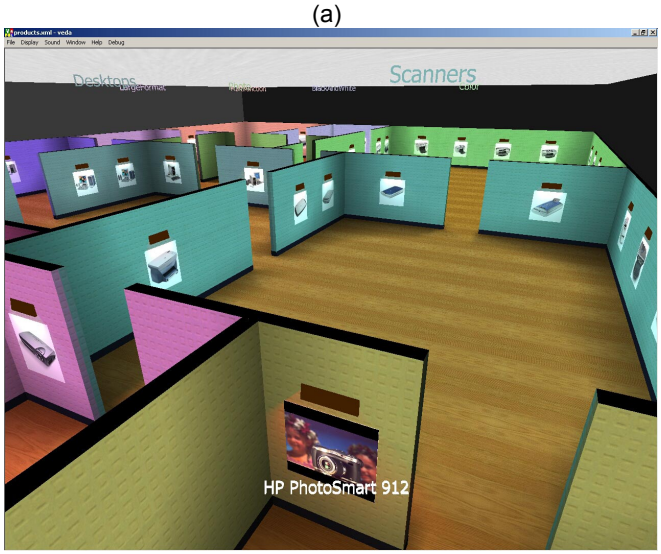
(c) (d)

**Figure 5. Environments automatically designed by VEDA for a movie collection based on (a)-(c) genre and (d) director.**

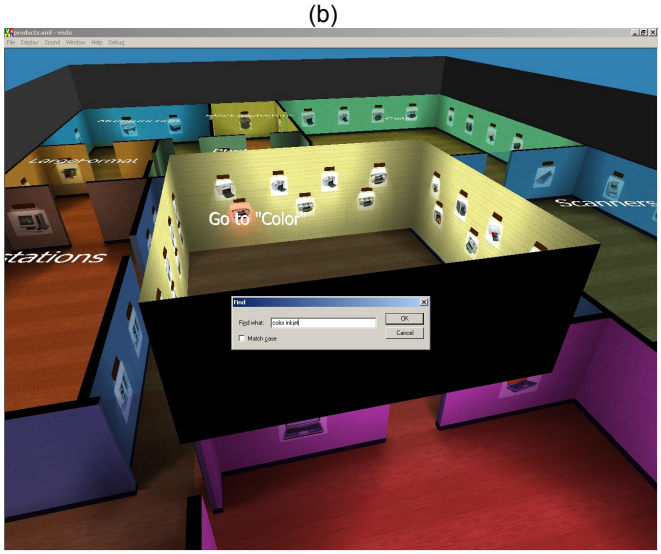

(a)




(b)




(c)




(d)

**Figure 6. Environments automatically designed by VEDA for a subset of HP products.**


(a) (b)

(c) (d)

**Figure 7. Environments automatically designed by VEDA for a digital photo collection.**