Managing Virtual Environments with Tree-Structured, Hierarchy-Embedded Virtual Objects

W M Rizhan W Idris[†], Md Yazid Mohd Saman[††], Aziz Ahmad[††], and Ahmad Shukri Mohd Noor[††]

[†]Faculty of Informatics, Universiti Sultan Zainal Abidin (UniSZA), 21300 Terengganu, Malaysia
[††]Faculty of Science and Technology, Universiti Malaysia Terengganu (UMT), 21030 Terengganu, Malaysia

Summary
Virtual reality (VR) refers to the use of computers and other related devices; and software to generate the world of simulation. Through VR, users are able to visualize, manipulate and interact with the computers and complex data to generate another world. It has been utilized in various applications in architecture, medicine, advertisement, business, entertainment, and education. However, developing VR environments is costly and expensive. Highly-technical persons are needed to create the virtual objects from scratch. Once a virtual system is created, managing and modifying it creates furher problems. There is a need for non-technical users to be able to create and modify their own virtual environments. This paper discusses a systematic and dynamic framework to manage virtual objects in virtual environment. It is called Virtual Reality System-Hierarchy Embedded Virtual Objects (VRS-HEVO). This VRS-HEVO framework comprises of two components; the Stand-Alone VRS-HEVO and the Distributed VRS-HEVO. They allow the virtual reality system to be implemented both in stand alone environment and distributed environment respectively. The models in VRS-HEVO include Data, HEVO and Viewing models. These are the basic models applied to both components. For Distributed VRS-HEVO, two other models are introduced; the Gallery and Client-Server models. They enable the VR system to be viewed in a distributed environment. To implement the models, object-oriented programming language was used. Java, Java 3D and Java Swing as the object-oriented programming languages and Socket programming are the main platforms in building the VRS-HEVO framework. For the usability and performance of the framework, virtual environments have been created to become as case studies. The tool has been percieved as an easy tool to use, especially for an environment in education.

Key words:
Virtual reality, Tree structure, Virtual environment, Virtual object.

1. Introduction

The use of computer technology especially in development of computer graphics has given a lot of contributions in implementing virtual reality (VR) systems. Many fields have utilized the VR systems such as in architecture, medicine, advertisement, business, entertainment and education [1]. The existence of applications or systems based VR has changed humans' perceptions, working styles and behaviors toward better effectiveness.

VR has been defined as the usage of computers and other devices; and software to generate the world of simulation [2]. Through the VR, users are able to visualize, manipulate and interact with the computers and complex data to generate another world [1]. The VR allows users to enter the computer-generated virtual world to interact with graphical objects and virtual agents with the sense of reality [3]. The users can also explore and navigate inside the virtual world as they walkthrough in actual environment. The term "Exploration" in the virtual reality is normally used to describe a user's displacement within a small area of space [4]. The combination of terms "Virtual Exploration" simulates walking or running short distances and "Navigation" is a term that refers to a long range travel such as spacecraft, aircraft or sea travel. Another term that often used in VR is "Walkthrough". The concept of "Walkthrough" means the interactive computer program can generate the experience of exploring or navigating in a building. It is meaningful especially for the purposes of visualizing and evaluating the models of buildings before they are constructed [5]. Artificial Reality, Cyberspace, Virtual World (VW), Virtual Environment (VE) and Synthetic Environment (SE) are the other concepts that also refer to the VR [6]. In fact there is a story behind the appearances of VW, VE and SE terms. Arns [7] has explained that these terms have come from an effort to move away the stereotypes from associating with the VR term. Most popular media have frequently misused or misunderstood the VR term by referencing to such activities such as 3D movies, video games and World Wide Web. Franchi [8] has described that VR is a computer that creates sensed experiences. The experiences can cause users to believe and differentiate between virtual experiences and reality experiences. The VR uses computer graphics, sounds and images to reproduce the electronically versions of real life situations. Although many definitions of the VR have been given, there are three essential principles of VR shared in any VR applications:

(i) Immersive - this refers to a situation that allows users to have feeling of being in a part of generated

Manuscript received December 5, 2010
Manuscript revised December 20, 2010

environment. Typically the involvement of interactions or manipulations in the system may lead to get better immersion. The capabilities of VR devices such as desktop computer, head mounted display (HMD) and wired gloves also contribute to the immersion.

(ii) Augmentation to the Real World - this refers to the perception towards the real world. It involves the integrations of virtual information or virtual models such as computer graphics, text, and sound in the physical environment so that users can perceive that information as existing in real-time [9].

(iii) Through Windows On World (WOW) - WOW is a VR system used to generate virtual environment.

Several types of the VR systems based on interface methods used in the system are Desktop VR, Video Mapping, Immersive VR, Telepresence, Augmented Reality and Fish Tank VR. There are five components in VR which are very important in VR applications.

(i) Virtual Object (VO) component refers to 3D model or 3D object that exist in virtual environment [4]. For examples: The models of people, animals, buildings and things.

(ii) Virtual Environment (VE) component refers to environment where models or objects exist [4]. The VE provides a 3D space to place the models or objects inside it. For examples: House and room.

(iii) Viewing component is visualization of a task [4]. All operations or tasks exercised in the VE can be visualized through the viewing component. This component can help explain the usage of terms; field of view; position and orientation; and distance of the objects. It is possible for the VR systems having several viewings or commonly pointed out as multi-views or multi-displays.

(iv) Walkthrough component is related to the interactive computer program to generate the walkthrough experience in a constructed building [5]. This component allows users to explore and navigate in the VE. The users may go forward and backward or may turn left and right as if they walk or run in the real environment. Typically, this component is used for the visualization and evaluation purposes in the construction, tourism and research fields [5].

(v) Manipulation component refers to the users' interactions with VOs in the VE [4]. Some examples of activities referring to this component are selecting an object, changing the position of an object, rotating and scaling an object.

Several examples of VR applications have used in various fields [1]; the Virtual Football Trainer application for virtual games; the Barcelona Pavilion, the Great Pyramid of Khufu and Detroit Midfield Terminal Project applications for virtual architectures; Medical Readiness Trainer (MRT) application for virtual medicals; Ship Motion Simulation, Virtual Prototyping of Automotive Interiors, Virtual Prototyping of a Concept Car, Virtual Simulation of Ship Production Processes, Accident Simulation, Maneuvering Submarine and Virtual Prototyping of Sailing Yacht applications for simulations and animations; Lake Michigan Flow Field Visualization, the Moebius Strip in VRML, Miller Indices in VRML applications for virtual visualizations; and Robots in Virtual Manufacturing for virtual Manufactures [10]; The other VR applications; WebSET project for medical training, Computer Augmented MRA System for medical visualization/surgical planning, Integrated Environment For Rehearsal and Planning Surgical Interventions (*IERAPSI*) for surgical simulation and the Virtual Tissues [11].

Although the achievements in the developments of VR applications have been tremendous, existing tools in creating VOs are costly to be purchased. They are also too technical to be used to create VOs from scratch. Sometimes, researchers find it inadequate to utilize the utilities provided in the 3D tools. The VE will be over loaded and filled with the undesired VOs if there is no proper management of the VOs. Indirectly such situation will lead to delivering information unsystematically.

In this paper, a framework to manage virtual objects in virtual environment is discussed in detail. It is called Virtual Reality System-Hierarchy Embedded Virtual Objects (VRS-HEVO). Tree data structure has been proposed as an approach to be used for managing the VOs in the VE. This provides a systematic and dynamic management of VEs.

2. Related Work

The components of VOs, viewing, walkthrough and manipulation used in the previous works have been studied. Dynamic Electronic Catalogs system have utilized pre-made files for creating VOs and VEs, walkthrough mechanism for navigation in the VE, rotation operation for manipulation of VOs and single-view mode for visualization [12]. In Virtual Community Trials Platform, a walkthrough method called Community 3D Walkthrough System has been proposed for moving, recording key walkthrough position nodes and playing back 3D animation with a walkthrough path. Single-view mode has been utilized for visualization [13]. However, there is no tree structure implemented in both VR systems. Some VR applications have implemented the tree structure for management of objects, data or information. In V-REALISM system, the tree structure is used to manage and manipulate the imported geometric models in

the VE. The models can be classified into environment model and system model in the tree structure. The environment models are used to simulate the maintenance environment and facilitate the visualization and exploration. The system models are further decomposed into component and subassembly models. Each component model is represented as a geometric mesh surface model which is displayed as a set of triangular surfaces. The smallest unit of a geometric model is triangular surface that is depicted as "TriUnit" in the hierarchy [14]. This system however can produce only a VE in a certain time. Yuan et al. [9] have proposed assembly guidance techniques using the Virtual Interaction Panel (VirIP) and the Visual Assembly Tree Structure (VATS). VirIP is used to track the real-time interaction pen using a Restricted Coulomb Energy (RCE) neural network. VATS is used to represent the assembly sequences of complex products. VATS specifically works for management of assembly information and retrieval for the AR-assisted assembly guidance system. VATS comprises nodes that represent the assembly information such as geometric properties of an assembly part, sub-assembly and so on. The sample tree proposed by Klein et al. [15] is used to store a number of polygons in a scene for rendering complex environment. The rendering process that provides caching mechanism allows only few nodes to be loaded from one frame to the next frame. This caching mechanism has employed the notion "Area Of Interest" (AOI) to use and render the objects in corresponding area. Meanwhile, Java 3D features have been used as main hierarchy structure in managing and creating the virtual anatomic models [16]. The scene graph concept has been used to represent the multi-modal anatomy models. Group node in the structure is similar to Java 3D scene graph *TransformGroup* node and Geometry nodes are composed by the geometry node in Java 3D with some functions needed by other modal data. Youn and Wohn [17] have implemented the tree structure called C-Tree in their collision localization algorithm. Their research focuses on localizing the possible collision regions by utilizing C-Tree for the 3D objects so that users can manipulate the 3D objects directly in real-time with the 3D glove-cursor.

3. VRS-HEVO Framework

A framework called VRS-HEVO is proposed to manage VOs in a VE. It describes the whole processes in the development of VR system for stand-alone environment and for distributed environment as depicted in Fig. 1. The framework can be split into two different designs as follows:

(i) Stand-Alone VRS-HEVO (SAVRS-HEVO) - This design utilizes the integrations of several models for developing VR system in stand-alone system generally and for managing VOs in VE specifically.
(ii) Distributed VRS-HEVO (DVRS-HEVO) - This design utilizes socket and gallery models to be integrated with Stand-Alone VRS-HEVO to produce the VR-based distributed system.

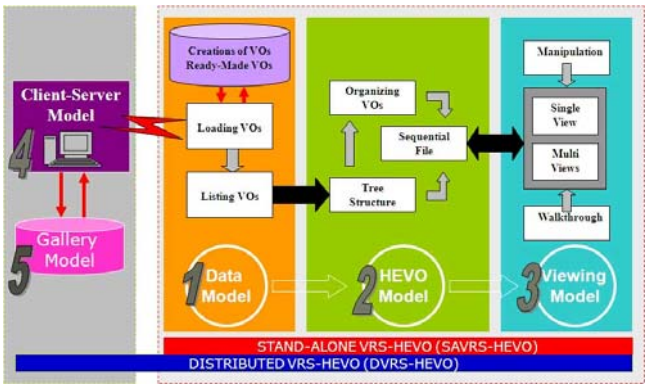


Fig. 1 Framework of VRS-HEVO

Three important models involved in the development of SAVRS-HEVO are Data Model, HEVO Model and Viewing Model. Data Model (DM) plays role as a location or database in a computer to place the VOs. Two important components in DM that correspond to the VOs are Loading Objects and Listing Objects. HEVO Model is particularly responsible to manage properly the loaded VOs in the VR system. For such purpose, Tree Structure, Organization of VOs and Sequential File components are utilized for directly contacting the VOs. Viewing Model (VM) is used to visualize VOs in VE. Four main components in VM are Single-View, Multi-Views, Walkthrough and Manipulation.

DVRS-HEVO design is executed to produce a VR system in distributed environment. This design technically can be divided into Server and Client sites. Socket programming plays an important role as a bridge that connecting between both sites so that sending and receiving data activities can be executed smoothly. In general, the server site comprises the Gallery Model and Client-Server Model. The client site contains all three models that have been discussed earlier in the SAVRS-HEVO design which are Data Model, HEVO Model and Viewing Model. Gallery Model acts as a place of resources or VOs in the server site. It functions as a database that contains a number of directories. Each directory represents a particular field and is named bases on its field. All virtual objects are collected in the directories that relevant to their fields. Client-Server Model (CSM) is a component that allows VRS-HEVO running in the distributed environment. For this case, socket is used as a bridge to connect the server and client sites. Communication,

sending and receiving data operations can be performed through the socket. The socket is also functioning to manage the simultaneous connections of many clients.

3.1 PetriNet Models of Distributed VR Environment

DVRS-HEVO involves multi-users or multi-clients working on the same data or sources for a certain task at their own preferred time. The simultaneous operations carried out in this environment may cause data inconsistency and deadlock in the system. It is crucial that the system is designed correctly so that data consistency and free-deadlock can be achieved in the system. For such purpose, a tool known as PetriNet (PN) is utilized [18]. This tool allows operations in the distributed environment to be simulated and animated before the implementation process begins. The PN models have been created using Platform Independent PetriNet Editor (PIPE2) software [19]. PN is a theory introduced by Petri [20]. PN is a graphical and mathematical tool to simulate and analyze systems that are sequential, concurrent, asynchronous, distributed and stochastic [21]. Fig. 2 shows a basic model of PN. It consists of five main components which are places, transitions, arcs, weight and marking.

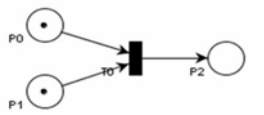


Fig. 2 Basic Model Of PetriNet

Mathematically, PN is defined as ***PN = (P, T, A, W, M_0)***, where:

$P = P1, P2, \ldots, Pm$ is a finite set of places
$T = T1, T2, \ldots, Tn$ is a finite set of transitions
$A < (P \times T) \cup (T \times P)$ = a set of arcs
$W : A \rightarrow 1, 2, 3 \ldots$ = a weight function
$M_0 : P \rightarrow 0, 1, 2, 3, \ldots$ = the initial marking
$P \cap T = \emptyset$ and $P \cup T \neq \emptyset$

Nodes and arcs are two important components in PN used to generate a graphical representation. The nodes may comprise place nodes (P) that are drawn as circles and transition nodes (T) as bars or boxes while arcs may act as connections between the place and transition nodes. The places may contain any number of tokens (black dots) which determines the state of a system. Tokens are moved from a place to another place by firing of transitions. Firing a transition is enabled if there is enough token in the input place.

In a distributed environment, the concurrency happens when two or more clients connects the server simultaneously to access the preferred data or VOs. Simulations of PN models for single user, two users and three users are presented.

Fig. 3 shows a PN model for the simulation of single-user accessing the shared resources in the DVRS-HEVO. Mathematically, Figure 3 demonstrates the PN model with a set of places, *P = {P1, P2, P3, P4, P5, P6, P7, P8, P9, P10}*; a set of transitions, *T = {T1, T2, T3, T4, T5, T6}*; a set of arcs, *A = {(P1,T1), (P5,T1), (T1,P4), (P4,T4), (P8,T4), (T4,P5), (T4,P9), (P9,T5), (T1,P2), (P2,T2), (P7,T2), (T2,P6), (P6,T5), (T5,P10), (T5,P7), (P10,T6), (T6,P8), (T2,P3), (P3,T3), (T3,P1)}*. Default weight function is one which is used to determine how many input tokens are needed to enable transition. The developed PN model for single user has been tested to be bounded, safe and free deadlock using PIPE2 [22].

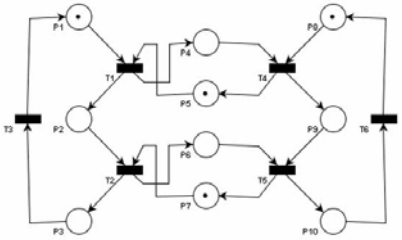


Fig. 3 A PN Model Of DVRS-HEVO For Single-User

Fig. 4 shows a PN model for the simulation of two users accessing the shared resources in the DVRS-HEVO. The PN model for such condition in mathematical is defined as a set of places, *P = {P1, P10, P11, P12, P13, P2, P3, P4, P5, P6, P7, P8, P9}*; a set of transitions, *T = {T1, T2, T3, T4, T5, T6, T7, T8, T9}*; and the initial marking of the system, M_0 *= (1,0,1,0,0,0,0,0,1,0,1,1,0)*.

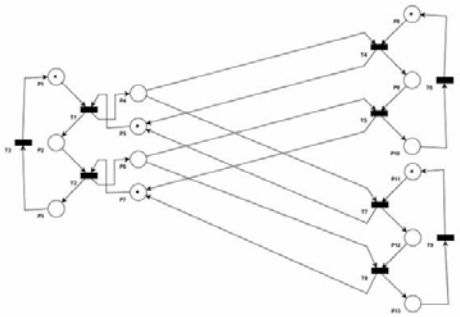


Fig. 4 A PN Model Of DVRS-HEVO For Two Users

A PN model for the simulation of triple or three users accessing the shared resources in the DVRS-HEVO is illustrated in Fig. 5. It demonstrates a set of places, *P = {P1, P10, P11, P12, P13, P14, P15, P16, P2, P3, P4, P5, P6, P7, P8, P9}*; a set of transitions, *T = {T1, T10, T11, T12, T2, T3, T4, T5, T6, T7, T8, T9}*; and the initial, M_0 *= (1,0,1,0,0,1,0,0,0,0,0,1,0,1,1,0)*.

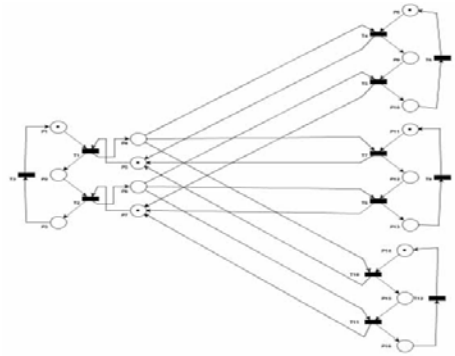


Fig. 5 A PN Model Of DVRS-HEVO For Triple-Users

4. HEVO Model

A hierarchical tree data structure has been proposed as a new approach to be used specifically for managing the VOs in the VE. This approach is called Hierarchy-Embedded Virtual Objects (HEVO). It is one model designed in the VRS-HEVO framework. In computer science, a tree is a widely-used data structure that emulates a tree structure with a set of linked nodes [23]. It is a hierarchically ordered, multi-level system that can be represented by an inverted tree. In the tree structure, each node may have parent node and zero or more children nodes.

The top node in the tree is called root node with the other nodes being connected below [24]. The very bottom nodes of the tree or the nodes that have no children are called leaves nodes (Fig. 6).

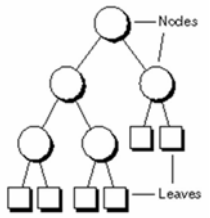


Fig. 6 Elements In A Tree Structure

Through mathematical definition [25], a tree T is a finite, non-empty set of nodes,

$$T = \{r\} \cup T_1 \cup T_2 \cup \ldots \cup T_n \quad (1)$$

With the following properties:

- A designated node of the set, r, is called the root of the tree; and
- The remaining nodes are partitioned into $n \geq 0$ subsets, $T_1, T_2, \ldots, T_n$, each of which is a tree.

For convenience to denote the tree T, the following notation is used,

$$T = \{r, T_1, T_2, \ldots, T_n\} \quad (2)$$

The other important terms [25] can be produced from **(2)**:

- The *degree* of a node is the number of sub-trees associated with that node. For example, the degree of tree T is n.
- A node of degree zero has no sub-trees. Such node is called a *leaf*.
- Each root r_i of sub-tree T_i of tree T is called a child of r. The term *grandchild* is defined in a similar manner.
- The root node r of tree T is the parent of all the roots r_i of the sub-trees T_i, $1 < i \leq n$. The term *grandparent* is defined in a similar manner.
- Two roots r_i and r_j of distinct sub-trees T_i and T_j of tree T are called *siblings*.

The tree structure in HEVO model allows the root node to become main VE. The sub-nodes in the tree structure may represent the other VEs or VOs. Basically, parent node may turn out as VE and child node may become VOs. The leaves nodes certainly represent as VOs.

In VR, the use of HEVO furnishes several advantages as follows:

(i) Categorizations of VOs
The VOs can be split into a number of categories or groups. The categorizations can be made when child nodes or sub-nodes are added or inserted under a particular parent node. In this case, the child nodes are said to be grouped or classified in their own parent nodes' category. The parent nodes that contain of their child nodes are defined as a collection of groups or categories.

(ii) Areas of Interests
Categorizing the objects will also produce a number of levels or depths. Fig. 7 shows the nodes in the tree structure that can be split into a number of groups based on parent-child concept. Each group then is called Area of Interest (AOF). For instances, Area 1 canvases the root node as parent while nodes 1, 2 and 3 as its children; Area 2 contains node 1 as parent node while its children are nodes 1.1, 1.2 and 1.3; and so do for the other areas. From the results, the VOs and VE can be defined and created successfully to be placed in the VR system. In this case, any parent node will be a VE while the child nodes will turn out to be the VOs in the VE (parent node).

(iii) Insertion of More Information
More details of information can be inserted especially for each VO in each area using HEVO technique. By separating the areas, all nodes that resident inside each area seem more obvious without much complexity. The VOs perhaps may get hold of complete descriptions to be displayed and delivered

as useful knowledge. Several main descriptions of a certain VO included name of VO, location, path, parent node, child nodes, number of child nodes and clarifications of VO.

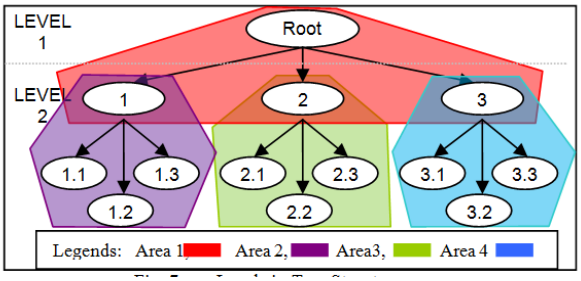


Fig. 7 Levels in Tree Structure

(iv) Modifications of Information and VOs
Any modification or change on either information or VOs especially in terms of their positions and orientations can be made independently and easily. The modification only affects at the particular infected area. The rest of areas are not influenced by the modification.

(v) Iterations of VOs
A VO in the tree structure can be reused or repeated using HEVO technique. There is no limitation in the duplication of VOs. The duplicated VO then can be manipulated to produce a unique VO that differs from the other VOs. For such purpose, a mechanism in HEVO to manipulate the VOs in terms of rotation, translation scale is used.

(vi) Synchronization of VEs
In HEVO, the VOs is managed and arranged in synchronized. A systematical flow of VOs' node can be seen in the tree structure from the root node till the leaves nodes. Each area of interest produced according to parent-child concept is managed in sorted. Therefore, it leads to the synchronization of VEs.

5. Development of VRS-HEVO

Four steps have been implemented to achieve the development of VRS-HEVO. They are shown in Fig. 8. In the first step, an interface of VRS-HEVO called Loading editor is initialized. This interface allows users to search the location of VOs either in users' computers or in the server and select the desired VOs to be loaded in the system; list the names of VOs that are successfully loaded; and display the sample of VOs selected from the list. The second step provides the use of data structure called HEVO editor. The users can create the graphical tree structure through several types of operations provided in this editor such as adding new VOs in the tree structure, adding new group of VOs, removing unneeded VOs and clearing all VOs in the tree structure.

Organizing VOs editor in the third step works to manage the VOs in the VE in terms of position and orientation. The users can view a VE in multi-monitor screens to precisely locate the VOs at the desired places. Each monitor screen projects different viewing such as front, left, top and perspective views. Through translation and scale operations, the users can move the selected VO at new location in the VE and resize VO. Information about VOs such as file, caption, description, parent, path, location, scale and rotation also can be saved systematically in the text file with extension **.hevo* for reference, distribution and display in future. Thus the users do not need to rebuild the VE from the beginning. The last step allows the users to view the final results of VEs either in single-view mode or multi-views mode. They can walkthrough in the VE using the arrow keys on the keyboard. The mouse is used to allow the users manipulating and interacting with VOs in the VE.

Begin
#1:Initializing interface of system → Loading Editor
i) Finding & Selecting VOs
ii) Listing Names of VOs
iii) Displaying Preview Screen of VOs
#2:Developing a data structure → HEVO Editor
i) Creating graphical tree structure
ii) Adding, removing and updating VOs
#3:Organizing VOs → Organizing VOs Editor
i) Multi-Monitor Screens
ii) Translation & Scale operations
iii) Recording VOs' Information
#4:Viewing VE → Viewing Editor
i) Single-View & Multi-Views
ii) Walkthrough in VE
iii) Manipulation of VOs
End

Fig. 8 Four Steps In The Development of VRS-HEVO

Based on the created models in the VRS-HEVO framework, four related editors have been built including Loading editor (Fig. 9(a)), HEVO editor (Fig. 9(b)), Organizing VOs editor (Fig. 9(c)) and Viewing editor (Fig. 9(d)). Java 3D have been proposed as an object-oriented (OO) programming language to be used as main platform in developing the VR system.

Loading Editor is an initial interface in VRS-HEVO. It functions to add or insert virtual object achieved whether from certain locations in the computer or from the database in the server. HEVO Editor is used to manage VOs using the hierarchy tree structure. Several types of operations such as adding new VOs, adding new group of VOs, removing unneeded VOs and clearing all virtual objects are provided in the HEVO Editor. Organizing VOs Editor works to manage the position and orientation

of each VO in the VE. For such purpose, multi-monitor screens are very important especially in precisely locating the VOs at the desired places. Each monitor screen projects different viewing such as front, left, top and perspective views. Two other activities available in this editor are translation and scale; and recording. The translation operation allows the selected VO to be placed at new location while scale operation enables the selected VO to be resized either to enlarge it or to make it small. The recording activity executes a sequential text file to record all information about each VO. The Viewing Editor functions to display the final products resulted from the activities that have been done through the Loading Editor, HEVO Editor and Organizing VOs Editor. Several activities available in this editor are users may switch at anytime either single view mode or multi views mode; users are able to navigate and surround in the VE; and users may pick and drag the scene graph objects to rotate the VO at 360 degree or to move the VO at somewhere.

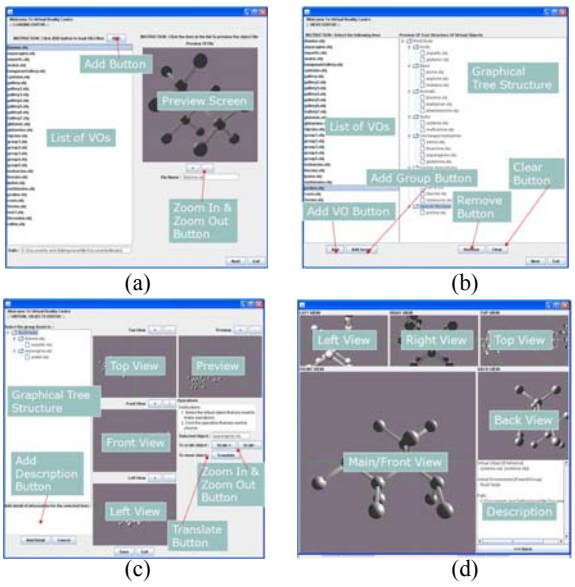


Fig. 9 Four Interfaces Of VRS-HEVO

6. Case Studies

In this paper, the results of several VEs visualizing World of Amino Acids, Virtual Forest and Underwater World are illustrated. VRS-HEVO has been used to develop them.

6.1. Case Study #1 – VE of Amino Acids

Amino acids are small molecules that may structure proteins [26]. By forming short polymer chains called peptides or polypeptides in turn, they construct proteins. In fact, the amino acids become the basic structural building units of proteins [27].

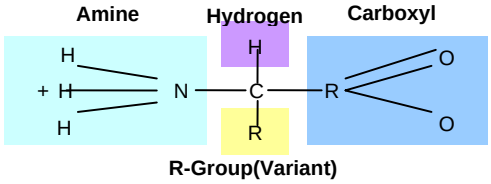


Fig. 10 The Structure of All Amino Acids

All molecules contain three important parts; an Amine group, a Carboxyl group and R group. Both the Amine group and the Carboxyl group belong to all amino acids. However, R group varies for individual amino acid. In fact, each molecule can be recognized through this R group. The Amine group contains elements NH3+ while the Carboxyl group contains COO- [26]. A different side chain is represented by R group specifically to each amino acid. Fig. 10 shows the basic structure for amino acids.

In using the HEVO method in this case study, several virtual environments of the amino acids can be created based on the classification of the amino acids. In this case, three layers will be obtained in the hierarchical tree structure. The first layer is where the root takes place. The root contains six child-nodes which refer to a Gallery Building and five classes of the amino acids i.e: *Polar with Positive Charge*, *Polar with Negative Charge*, *Polar Uncharged*, *Non-Polar* and *Aromatic*. To develop the model of each group, *new group* utility in the HEVO system is used. The name of new group then is changed based on the classes of amino acids. It is because the group has no model to represent itself in the virtual environment. In the last layer, all children or virtual objects will be listed according to their groups respectively.

Fig. 11 depicts the tree structure for the amino acids group using HEVO method. Through the tree structure in HEVO, the VOs and VE components are able to be identified. In this case, all nodes that have been parents in each group become the VEs which are inhabited by their own VOs. Meanwhile, the VOs refer to the contents of the parents' children. Therefore the root and the groups have been determined to be the VEs.

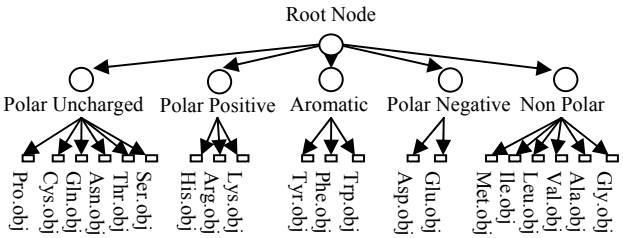


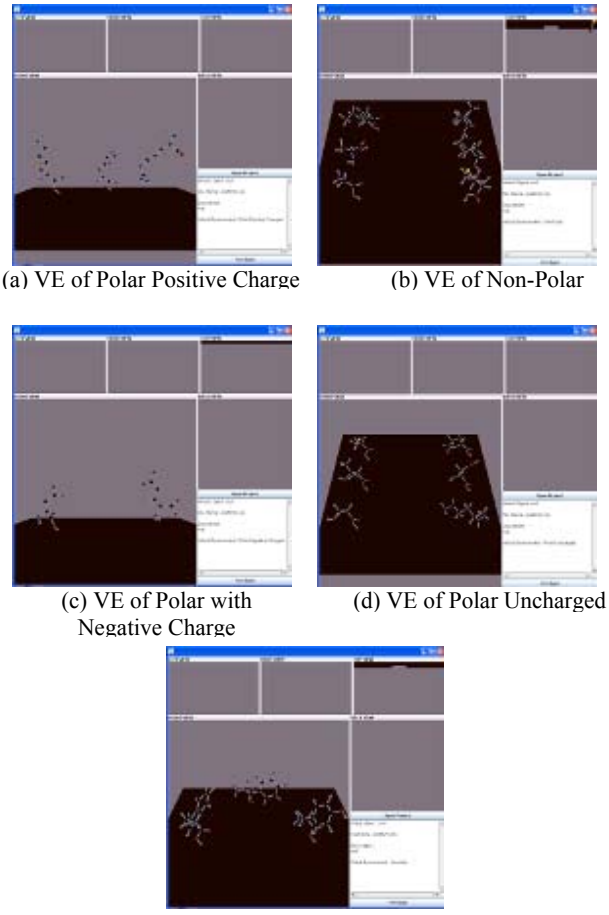
Fig. 11 Tree Structure of Amino Acids Group

Walkthrough and manipulation utilities provided in HEVO are able to be performed in each defined level. The walkthrough operation functions as a navigation tool in the published VEs. This tool helps users navigate from one place to another place in the 3D environment. Interactions with the virtual objects can be implemented as well through the manipulation operation. This useful tool allows users to gain details of information of the interacted VOs and to transit from one VE to another VE when they interact directly with certain VOs.

As illustrated in Fig. 12, there are a Gallery and five other categories of amino acids in a VE. These categories are placed inside the Gallery as if they are exhibited in it.

Fig. 12 A Gallery And Five Main Categories

These five categories in the Gallery refer to Polar Positive Charge, Polar Negative Charge, Non-Polar, polar uncharged and aromatic. Each of them has its own sub-nodes or child nodes. Thus the categories can have their own VEs when user activates them. Fig. 13(a), Fig. 13(b), Fig. 13(c), Fig. 13(d) and Fig. 13(e) show the VEs for each category and their VOs.



(a) VE of Polar Positive Charge (b) VE of Non-Polar

(c) VE of Polar with Negative Charge (d) VE of Polar Uncharged

(e) VE of Aromatic

Fig. 13 VEs For The Categories of VOs

6.2 Case Study #2 - Virtual Forest

The case study #2 simulates a virtual forest using VRS-HEVO. A forest is an area with a high density of trees [28, 29]. The virtual forest can be made by clustering the trees. In this case, four types of trees have been used to form the virtual forest. As shown in Fig. 14, the trees are utilized iteratively under the root node in the tree structure. Many trees will form a forest. Even the forest takes the same tree to be clustered, using scale, translate and rotate tools in the system may adjust the tree to be uniquely distinguish between each others.

For this VE, four basic VOs of trees have been used iteratively to construct the full virtual forest. Using the utilities in the VRS-HEVO, the trees look uniquely distinguishes from each other. The snapshots of the virtual forest have been demonstrated in Fig. 15.

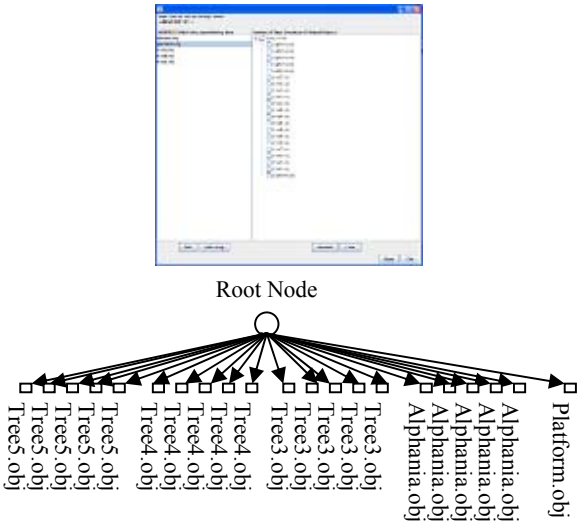


Fig. 14 Tree Structure of Virtual Forest

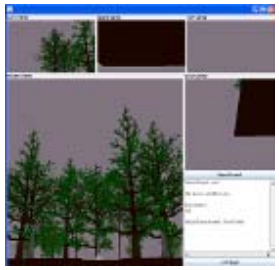


Fig. 15 Virtual Forest In VRS-HEVO

6.3 Case Study #3 – Underwater world

Underwater world describes the realm below the surface of water in a natural feature called a body of water such as an ocean, sea, lake, pond or river [30].

Using VRS-HEVO, a VE of underwater sea is developed and presented. A collection of ready-made VOs as shown in Fig. 16 is compiled. The ready-made VOs are downloaded freely over the Internet. To bring realism in the underwater sea, the platform containing water and sand surface has been created. In VRS-HEVO, the VOs are compiled in the graphical tree structure as depicted in Fig. 17. All VOs are inserted under the root node. The snapshots of the virtual underwater sea have been demonstrated in Fig. 18. In the VRS-HEVO, all creatures including whales, sharks, dolphins and plants are collected in virtual ocean to bring realism and live.

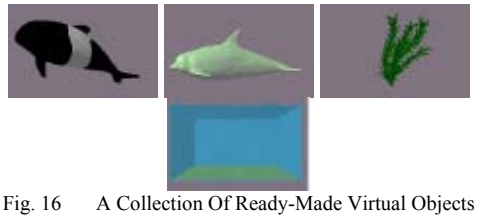


Fig. 16 A Collection Of Ready-Made Virtual Objects

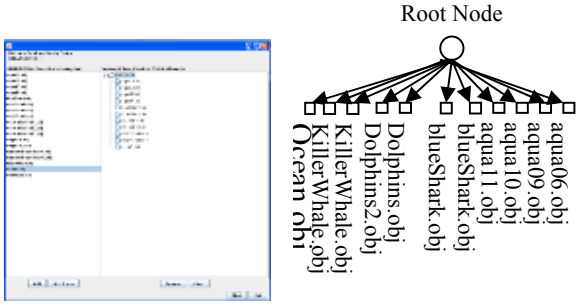


Fig. 17 Tree Structure of Virtual Underwater Sea

Fig. 18 Virtual Underwater Sea In VRS-HEVO

7. Usability Study Of VRS-HEVO

A usability study on the VR system in education has been conducted to a group of undergraduate students. The evaluation of VRS-HEVO would ensure that the effectiveness, efficiency and usability existed and performed in the system as desired. Inconvenience and difficulty encountered by the users during interactions can be triggered and considered as normal reactions. Identified factors for such problem can be used to improve the system or can serve as references to design and develop the future systems. The virtual environment of the Amino Acids has been used. The experiment has two sample quizzes for the students to familiarize with. Later, they tested the system and took a real quiz. After the completion of the quiz, they filled up a questionnaire form to express their experiences and to evaluate the performance of the VRS-HEVO.

Fig. 19 shows the mean scores of the forty two responses for the usability study of the system. Most of the respondents state their agreements that the system can help them in education. They have an opinion that that the

system is easy to use. However, there is a need for them to learn and familiarize with the system before use. Based on the findings, the system has shown to be:

- A useful tool for teaching.
- Easy to use.
- Used frequently.

The respondents have responded that:

- They felt fun using the system.
- They can understand the subject using the system.

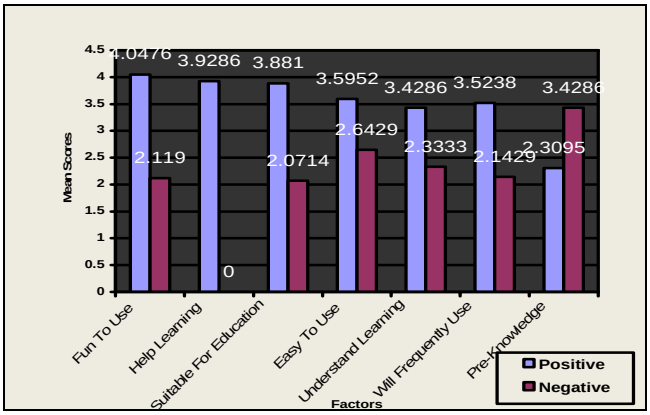


Fig. 19 Usability Study of VRS-HEVO

8. Conclusion

This paper has presented a framework of VRS-HEVO for managing VOs in VE. Though VRS-HEVO, users are able to visualize, manipulate/interact with VOs and navigate/walkthrough in the VE. Two designs introduced in the framework are SAVRS-HEVO for stand-alone environment and DVRS-HEVO for distributed environment. Several models that support both types of functionalities have been discussed. To develop the VRS-HEVO in stand-alone system, the Data Model, HEVO Model and Viewing Model have been used. DVRS-HEVO has been used to create a VR system for a distributed environment. They are modeled as the Gallery Model and the Client-Server Model.

A qualitative method has been used to evaluate the usability of VRS-HEVO. The findings have shown that the VRS-HEVO is useful for teaching, easy to use and being used frequently in the future. The respondents have also indicated that they have fun to use and comprehend the subject using the system.

This approach may breed better understanding among students in their subjects. For future works, this VR field may offer many potential applications included tourism, gaming, manufacture, architecture, medicine, advertisement and entertainment. Moreover not many researches previously issued the capability of VR systems that implement managing VOs function in the Internet whether in applet and html.

Acknowledgment

The authors would like to express their appreciations for the fundings from UniSZA and UMT.

References

[1] Grady S.M., Virtual Reality: Computer Mimics The Physical World, Fact On File Inc, New York, 1998.

[2] Eddings J., How Virtual Reality Works, Emeryville, Ziff-Davis Press, 1994.

[3] Youn J. H., & Wohn K., Realtime Collision Detection For Virtual Reality Applications, IEEE-Virtual Reality Annual International Symposium, No. 9, pp. 415-421, 1993.

[4] Parent A., Visual Information Technology. Life-like Virtual Environments: An Introductory Survey Applications and Activities, Design Requirements and Guidelines, ERB-1055, NRC No. 41555, 1998.

[5] Funkhouser T.A., Sequin C.H., & Teller S.J., Management Of Large Amounts Of Data In Interactive Building Walkthroughs, Symposium on Interactive 3D Graphics, Proceedings of the 1992 symposium on Interactive 3D graphics, Cambridge, Massachusetts, United States, pp. 11 – 20, ISBN:0-89791-467-8, 1992.

[6] http://www.ftsm.ukm.my/samn/PDFs/TH3813%20 01%20Pengenalan.pdf, Accessed on November 3, 2009.

[7] Arns L. L., A New Taxonomy For Locomotion In Virtual Environments, Ph.D Dissertation, Iowa State University, 2002.

[8] Franchi J., Virtual Reality: An Overview, Journal of TechTrends, Springer Boston, Volume 39, Number 1, 1994.

[9] Yuan M.L., Ong S.K., & Nee A.Y.C., Assembly Guidance in Augmented Reality (AR) Environments Using a Virtual Interactive Tool, International Journal of Production Research, Vol. 46, No. 7, pp. 1745-1767, 2006.

[10] http://www-vrl.umich.edu/projects.html, Accessed on Mei 7, 2009

[11] Leng J., Scientific Examples of Virtual Reality and Visualization Applications, UK High Performance Computing, pp. 1-13, 2001.

[12] Benjamin P. C. Yen., & Kenny Y. M. Ng., Development and Evaluation of Dynamic Virtual Object Catalogs, Information And Management, No 40, pp. 337-349, 2003.

[13] Qiang L., Huanzhi L., Fei W., & Boyan C., Virtual Community Trials Platform, Journal Of International Archives Of Photogrammetry Remote Sensing And Spatial Information Sciences, Vol 35, (4), pp. 1144-1147, 2004.

[14] Li J. R., Khoo L. P., & Tor S. B., Desktop Virtual Reality For Maintenance Training: An Object-Oriented Prototype System (V-REALISM), Elsevier-Computer In Industry, 52, pp. 109-125, 2003.

[15] Klein J., Krokowski J., Fischer M., Wand M., Wanka R., & Heidi F. M. A. D., The Randomized Sample Tree: A Data Structure for Interactive Walkthroughs in Externally Stored Virtual Environments, MIT Press Journal, Vol. 13, No. 6, pp. 617-637, 2004.

[16] Huang S., Baimouratov R., & Nowinski W. L., Building Virtual Anatomic Models Using Java3D, Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry, Session: 7-3 Modeling, pp. 402 – 405, 2004.

[17] Youn J. H., & Wohn K., Realtime Collision Detection For Virtual Reality Applications, IEEE-Virtual Reality Annual International Symposium, No. 9, pp. 415-421, 1993.
[18] Petri Net, http://en.wikipedia.org/wiki/Petri_net, Accessed on Mei 7, 2009
[19] PIPE2, http://pipe2.sourceforge.net/, Accessed on Mei 7, 2009
[20] Petri C. A., Kommunikation Mit Autometan. Technical Report, Doctoral Thesis, University of Bonn, 1962.
[21] Fadhilah A., A Framework of Integrated Model Base for Decision Support System, Dissertation of Doctor of Philosophy, Universiti Malaysia Terengganu, 2009.
[22] W M Rizhan W. I., A Framework For Managing Virtual Objects In Virtual Environment, Dissertation of Master of Science, Universiti Malaysia Terengganu, 2010.
[23] Tree Data Structure, http://en.wikipedia. org/wiki/Tree_data_structure. Accessed on April 10, 2008
[24] Goodrich M. T., & Tamassia R., Data Structure And Algorithms In Java, John Wiley & Sons Inc, USA, 1998.
[25] Preiss B. R., Data Structure And Algorithms With Object-Oriented Design Patterns In Java, John Wiley & Sons, Inc, USA, 2000.
[26] Millar T., Biochemistry Explained: A Practical Guide To Learning Biochemistry, Gutenberg Press Ltd, Malta, 2002.
[27] Polanski A., & Kimmel M., Bioinformatics, Springer-Verlag Berlin Heidelberg, New York, 2007.
[28] Forest, http://en.wikipedia.org/wiki/Forest, Accessed on October 9, 2009.
[29] Kalman B., & Smithyman K., What Is A Forest?, Crabtree Publishing Company, United Kingdom, 2003.
[30] Underwater, http://en.wikipedia.org/wiki/ Underwater, Accessed on November 2, 2009.