UTILIZING A 3D GAME ENGINE TO DEVELOP A VIRTUAL DESIGN REVIEW SYSTEM

Mohd Fairuz Shiratuddin, Ph.D.
School of Information Technology, Murdoch University, Murdoch, WA 6150, Australia
f.shiratuddin@murdoch.edu.au

Walid Thabet, Ph.D.
Department of Building Construction and Myers-Lawson School of Construction,
Virginia Tech, Blacksburg, VA 24601, USA
thabet@vt.edu

SUMMARY: A design review process is where information is exchanged between the designers and design reviewers to resolve any potential design related issues, and to ensure that the interests and goals of the owner are met. The effective execution of design review will minimize potential errors or conflicts, reduce the time for review, shorten the project life-cycle, allow for earlier occupancy, and ultimately translate into significant total project savings to the owner. However, the current methods of design review are still heavily relying on 2D paper-based format, sequential and lack central and integrated information base for efficient exchange and flow of information. There is thus a need for the use of a new medium that allow for 3D visualization of designs, collaboration among designers and design reviewers, and early and easy access to design review information.

This paper documents the innovative utilization of a 3D game engine, the Torque Game Engine as the underlying tool and enabling technology for a design review system, the Virtual Design Review System for architectural designs. Two major elements are incorporated; 1) a 3D game engine as the driving tool for the development and implementation of design review processes, and 2) a virtual environment as the medium for design review, where visualization of design and design review information is based on sound principles of GUI design. The development of the VDRS involves two major phases; firstly, the creation of the assets and the assembly of the virtual environment, and secondly, the modification of existing functions or introducing new functionality through programming of the 3D game engine in order to support design review in a virtual environment. The features that are included in the VDRS are support for database, real-time collaboration across network, viewing and navigation modes, 3D object manipulation, parametric input, GUI, and organization for 3D objects.

KEYWORDS: collaborative, design review, game engine, torque, virtual environment

1. INTRODUCTION

In many industries, the main objective of design review is to make sure that the priorities of the project are met. If design review is properly performed, it provides a mechanism whereby the total design activity can be carried out in an objective manner, leading to improved designs and products (Pugh, 1991). In the Architecture, Engineering and Construction (AEC), the design review process allows design and design documents to be periodically checked for inconsistencies, errors and other aspects of design which does not concur with the owner's requirement (East, 1998), prior to the actual construction.

Design review is still imperfect despite the current available tools because reviewers are subject to many pressures in the design review process (East, 1998). Design reviewers are assigned to "more critical" design and construction tasks, thus creating backlogs of un-reviewed drawings and specifications. This may force reviewers to sacrifice the thoroughness of their reviews. Also, time constrains on project funding cycles restricts the ability of the designer to incorporate changes that have been recommended by design reviewers. As a result, jobs may be bid with known mistakes. It can be concluded that the potential of design review is promising but current methods are still deficient thus causing the benefits to be unattainable.

In a typical project, design reviews are performed in four consecutive stages at the end of each design phase: Programming, Schematic, Preliminary and Working Drawings (Spillinger 2000; East *et al.* 1995; Fu & East 1999; Shiratuddin, 2009). 2D CAD drawings (in digital or paper form) and specifications are the documents that are heavily used and referred to during design review. These documents are prepared by various specialists such as the architect, electrical engineer, structural engineer, mechanical engineer, etc. Thus, coordinated and error-free design documents are usually difficult to achieve. For example, a designer produces designs in 2D format (hand or computer-aided) which is derived from his 3D mental models and converted into 2D designs and information. In addition to that, 2D drawings do not adequately represent the spatial information and requirements of the building components. The process of interpreting these designs by other project team members is undoubtedly influenced by their varying individual experiences. For that reason, no two persons' interpretation and visualization of the designs may be the same. This is one of the causes for errors and conflicts in designs, and becomes the basis for miscommunication throughout the project lifecycle. The design review process therefore attempts to minimize such problems.

The current methods of design review lack a central and integrated information base to make the flow of information more efficient. Information is still at large, spread across various places in electronic and physical paper forms such as construction documents, reference books and guides, manuals, building codes and regulations, and etc. (Shiratuddin, 2009). Some of the methods used for design review include the use of light tables and manually cross-checking the designs, the use of physical and electronic checklists, and the use of physical mock-ups (PMUs). Overlaying 2D drawings on a light table is too time-consuming and inefficient (Staub-French & Fischer, 2001). Both the light table and the interdisciplinary checklist (Nigro & Nigro, 1992) methods rely heavily on paper-based formats, and require manual cross-checking of drawings, specifications and other documentations. The use of web-based online review system (Nigro & Nigro, 1992; East *et al.*, 2004) involves perusing web pages, which is to some extent comparable to paper-based format, only substituted electronically on the computer with some information storage and retrieval capabilities. Using PMUs are too expensive and time-consuming and is inefficient in terms the handling of information (Shiratuddin, 2009).

There is thus a need for design review related information to be centralized, easily accessible and processed to assist design reviewers during design review. East *et al.* (2004) suggested that a successful automated design review system will reduce the number of design reviewers and the time required to complete the reviews, thus, shorten the project delivery cycle. On time or earlier project completion will provide the owner with favorable earlier occupancy, which then translates into significant total project savings to the owner. Therefore, there is a need to represent designs not just in 2D but in 3D, where designers, reviewers, and other project team members can view and collaborate using the same 3D model. Designers can also be sure that the designs they produce is what the owner envisioned. Utilizing virtual environment (VE) technology can improve the 3D representation of the design and

provide a common language for the project stakeholders. This level of collaboration ensures that everyone impacted by the design will have early access to the design information. It also provides the ability for project team members at an earlier stage if design, to influence the decision made for the final design. A means for real-time collaboration within the same 3D space is therefore needed in the design review process.

In this paper, we present our work on the development of a design review system, the Virtual Design Review System (VDRS) for architectural designs. The VDRS was developed utilizing the Torque Game Engine (TGE). The VDRS incorporated two major elements; 1) 3D game engine as the driving tool for the development of the VDRS and implementation of design review processes, and 2) a virtual environment (VE) as the medium for design review, where the visualization of design and design review information is based on the principles of GUI design. The development of the VDRS involves two major phases; firstly, the creation of the assets which includes a 3D model of a three bedroom house with textures, and the assembly of the VE. Secondly, the modification of existing functions or introducing new functionality through programming of the TGE in order to support design review in a VE. The features that are included in the VDRS are support for database, real-time collaboration across network, viewing and navigation modes, 3D object manipulation, parametric input, GUI, and organization for 3D objects.

2. DESIGN REVIEW IN THE AEC INDUSTRY

The term "design review" used in this paper refers to the reviewing of drawings which are prepared by the designer/architect/engineer during the different stages of design; programming, schematic, preliminary and working drawings. In the AEC industry, design review is a process in which information is exchanged between the designers and reviewers to resolve any possible design related issues and to accommodate the interests, goals, and objectives of the owner, in a timely manner. The process ensures the quality of designs (East, 1998) that includes the materials to use; the building systems components (structural, mechanical, electrical, plumbing etc.); compliance with the current codes and standards; and the maintenance requirements.

Fu and East (1999), and Shiratuddin (2009) describe the typical process for conducting design reviews includes three steps: (1) The reviewer examines the plans and specifications for a project; (2) the reviewer, drawing information from experience, standard references, or from other available sources, notes down the potential design errors or omissions found; and (3) the reviewer writes down a list of review comments to be returned not only to the designer, but also shared with other reviewers. The drawings are given back to the A/E for redesign and the incorporation of the required changes. The process of design, review and redesign is cyclical in nature until satisfactory drawings are accomplished and approved. Once approved, the drawings for that particular design phase are considered complete. The A/E can start with the consequent design phases until the drawings are finally ready to be used for construction. The project manager or design review coordinator ensures the right flow of documents and communication among the parties involved. Due to time and resource constraints imposed on the reviewers, the entire design review process is regarded as tedious and intensive (East *et al.* 1995; Ichida 1996; Fu & East 1999; Staub-French & Fischer 2001).

3. THE VIRTUAL DESIGN REVIEW SYSTEM (VDRS)

The AEC industry is shifting more and more toward utilizing Building Information Model (BIM). Many software vendors such as Autodesk, Graphisoft, Tekla, Vico Software etc. are developing and continue to improve features that supports the principles of BIM specifically in the areas of information interoperability and information reusability. However, many of these software focus more on design, and not design review. The Virtual Design Review System (VDRS) differs from a variety of BIM software presently available to the AEC industry. The VDRS specifically emphasizes on the design review portion of design which takes place during the design stage of the building construction life-cycle. The VDRS supports the idea of a more methodical, streamlined and collaborative approach to design review. A distinct feature of the VDRS is that it allows multiple users to collaboratively review design in real-time, and in a 3D virtual environment (VE). Users exist within the VDRS as avatars in the same time and virtual space. In the VDRS, synchronous collaboration occurs via text chat, voice communication, and through interaction with shared design components. Current BIM software does not allow for such real time synchronous collaboration. Within BIM software however, users may collaborate asynchronously, through text annotation and the marking up of designs via whiteboard-type features (Shiratuddin, 2009).

In this paper, we present the development of a prototype Virtual Design Review System (VDRS) utilizing a 3D Game Engine. The VDRS was developed using the Torque Game Engine (TGE) from GarageGames. Using C++ programming and C-like syntax scripting languages, additional codes were written to provide the added functionalities specific for design review. Microsoft Visual Studio 2008 was used as the programming IDE (integrated development environment) and compiler, installed on an IBM-PC compatible desktop computer. The VDRS supports VE devices such as the head-mounted-display (HMD), tracking devices, data glove, 3D navigation input devices, game input devices and stereoscopic display (through the use of nVidia's consumer stereo display driver).

Figure 1 shows the building blocks of the VDRS and how they are linked to one another. The TGE provides the real-time 3D visualization in the VE, the graphical user interface (GUI), and the capability for 3D object manipulations. Whenever a design reviewer interacts with the 3D objects and information in the VE, interactions among the building blocks happen in the background. The interaction between the database engine and rule-based engine is almost cyclical. When the requested information is retrieved (based on the flagged *True* statement provided by the rule-based engine), the information is then passed to the TGE so that information can be visually presented to the design reviewer in the VE.
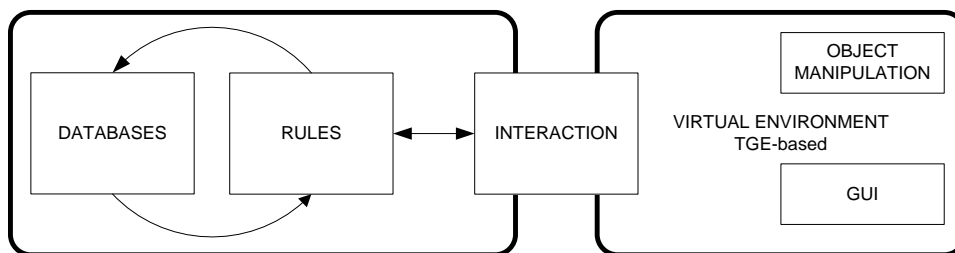

FIG 1: The building blocks of the VDRS (Shiratuddin, 2009)

The databases store the 3D models, checklists, and sources of design review related information which can be accessed by the design reviewer at a click of a button in the VE. Access to design review related information is benefited from a defined intelligent search and retrieval processes. The intelligent search and retrieval processes include filtering, querying and sorting the required design review information which is processed by the "Rules" block (Figure 1). The processes are based on four information processing contexts: Discipline, Tasks, Objects, and Location. Whichever contexts triggered or selected by the design reviewer, and once the 3D model in the VE has been reviewed, if any errors or inconsistencies are encountered, the design reviewer can quickly access any related

design review information stored in the databases for validation. After verifying the design issues, the design reviewer can then input any comments or recommendations into the VDRS. The comments and recommendations are then stored in the respective database that are accessible to other design reviewers, and most importantly to the designer who is responsible to make the design revisions.

3.1 The 3D Game Engine as a Tool for the development of the VDRS

Computer games that include 3D virtual environments use 3D Game Engines to produce and display images in real-time on the display device. Unlike 3D engines, these 3D game engines include varieties of features that are required in game environments, such as sophisticated real-time 3D scene rendering, networking, physics, artificial intelligence, scripting and many more (Finney, 2007). Each game uses the different features to organize how the visual aspects of the game be modeled and presented on screen. At present, these visual aspects have become increasingly important as games are becoming more focused on 3D VE, richness in textures and forms, overall realistic feeling, playability and user involvement (immersion) with the game.

Strengths of a 3D Game Engine

The strengths of a 3D game engine that lies within the VDRS follows these criteria: real-time rendering, real-time walkthrough, interactivity, multi-participatory feature, lighting and collision detection.

Real-Time Rendering

Adding visual characteristics such as shading, shadows, and textures make 3D models more visually realistic. The entire process of calculating the appearance of the 3D model, converting it to an entity that can be drawn on a two-dimensional (2D) screen and then displaying the resulting image is called *rendering* (Finney, 2007). Rendering is the process of converting the 3D mathematical model of an object into an on-screen 2D image. There are various techniques to apply color to (or rendered on) the faces of an object such as Flat Shading, Lambert Shading, Gouraud Shading, Phong Shading, Fake Phong Shading, Texture Mapping, Shaders, Bump Mapping, Environment Mapping, and Mip-mapping (Finney, 2007).

Real-time rendering also measures the performance of a 3D game engine. Mullen (1998) suggested benchmarking the performance of a 3D game engine running on a computer can be done by measuring the number of images generated on-screen per second i.e. frames per second (fps). Normally, when a 3D model is developed, the complexity increases as the level of realism increases. This will increase the real-time rendering time, and can drop the fps of a real-time walkthrough quite noticeably. When the fps drops from 30 to between 5-10 fps while displaying a 3D model, the situation is no longer acceptable for presentation purposes hence making inspection of the 3D model disorienting and difficult (Miliano, 1999). Many modern 3D game engines can maintain an interactive rate of at least 30 fps while handling a large number of polygons in a single scene that includes lighting and texturing.

Real-Time Walkthrough

A game-based VE has a real-time walkthrough feature that allows design reviewer to feel as though "they are there", walking through space, able to move upstairs, peering out windows, and etc., which in turn also give them sense of scale. Realism and details in a VE are achieved through the process of adding 3D qualities such as shadows, colors and shade variances. According to Campbell and Wells (1994), such are the criterion that makes a VE closer to reality because of the ability to allow "immediate, direct, and more intuitive control over a three-dimensional design". Not only the reviewer benefits, an owner of a project can also freely inspect a virtual facility beforehand and can better set realistic expectations on the final product, rather than just viewing representations in the forms of 2D drawings, static image rendering or fixed-path animation (Shiratuddin & Thabet, 2003).

Interactivity

The term interactivity refers to the interaction between a computer and a user which takes place through the changes of location views, typed commands, voice commands, mouse movements and clicks, or other means of interfacing. The game engine accepts and responds to the user's activity in real-time at an interactive rate of at least 30 fps. Interactive features are important for the users of the VDRS as it will relate what they are seeing in the VE to the real world. This feeling of realism is important to convince the users that the environment is realistic and represent the real world (Mays, 1998; Miliano, 1999).

Interactive controls in the VDRS are used either by clicking on them or click-dragging the mouse across them. Some controls such as the edit boxes, also require a user to type in text using the keyboard. Some of the controls have built-in labels that identify their purpose, and some will require the developer to create an accompanying non-interactive control with a custom label. In the VDRS, non-interactive controls, as the name implies, are used to display information and not to capture user input.

Multi-Participatory

In a VE, avatars are geometries (Vince, 1998) that can have intelligent characteristics (i.e. AI characters or bots) or simply a virtual representation of the user than can be controlled by the user's input. The VDRS is also a collaborative and distributed VE which allows for multiple design reviewers to perform collaborate design review in real time within the same time and virtual space. Such synchronous collaboration can occur via text chat, voice communication and through interaction with shared design components. Current 3D CAD systems do not allow for such real time synchronous collaboration but there are possibilities for users to collaborate asynchronously. Typically this is done through text annotation and the marking up of designs via whiteboard-type features.

Lighting

Many 3D game engines incorporate varieties of lighting features that resembles real world lightings. Lighting is important as it can provide the sense of security and confidence when users are occupying or maneuvering an enclosed space such as walking through a building. There are two types of lightings; dynamic and static. A dynamic light can be seen in the games environment when a weapon is fired where the blast of fire can cast light off the walls and surrounding objects, or when the user's avatar is casting shadow. Using dynamic lights can be expensive hence can reduce the fps of a VE as they cast real-time shadows. Static lighting, on the other hand, does not move, and usually used to provide overall lighting condition in a VE. Unlike dynamic lights, static lights do not always cast shadows in real-time hence they are more efficient and can increase the fps of a VE. The VDRS supports dynamic and static lighting, but since the design review process does not review lighting conditions, do lighting analysis or study, no lights were used in the VDRS's virtual environment.

Collision Detection

Collision detection is described as the process of detecting when two or more objects in a VE come into contact with one another (Maurina, 2006). Collision detection improves interactivity in a VE. Many current commercial VE tools, collision detection has to be manually defined by the designer during the scene-building process. By default, most 3D game engines enables collision detection for solid objects therefore disallowing users from walking through walls, doors etc. Many 3D game engines automatically detect when user collides with solid or non-solid objects, thus giving users the experience of 'bumping' as in real life (Shiratuddin & Thabet, 2002). The VDRS employs collision detection by default to any objects that are considered solid such as walls, doors, windows etc. Design reviewers are able to turn off collision detection feature on-the-fly should they wish to review the 3D model without it.

In summary, the continuing popularity of computer and video games has instigated the development and introduction of brand names 3D game engines. The innovation and technological advancement of many current 3D game engines has allowed for not only games to be developed, but also used for serious real-world applications development such as the VDRS. However, prior to delving into utilizing a 3D game engine for developing such applications, one must

carefully considers the licensing cost if the final intended application will be a commercial product. The cost of licensing of these engines can range from no upfront cost at all (hence free) to a substantial financial investment. Some examples of free Open Source 3D game engines that are currently available are Crystal Space, Ogre, Irrlicht and Panda 3D. Some 3D game engines are also affordable and meant for independent (indie) developers or small companies, and they include the Terathon's C4Engine, GarageGames Torque Game Engine, and Unity Technologies Unity engine. There are also 3D game engines that can cost a fortune to license for large scale commercial purposes such as the Epic Games Unreal Engine, Valve Half-Life and Crytek Crysis.

3.2   The Torque Game Engine (TGE)

The Torque Game Engine (TGE) was originally conceived by a computer game development company called Dynamix. Dynamix designed and developed computer games titles such as Earthsiege, Starsiege, Tribe and Tribes 2 (Maurina, 2006; Finney, 2007). The founders of Dynamix then founded GarageGames and currently distribute the TGE as one of the products for independent (indie) game developers. The low-cost licensing option has been attractive enough for many indie developers to use the TGE for game development purposes. At the time of this research, the TGE was purchased based on an indie license for $150. GarageGames' business model is somewhat different from many other 3D game engine developers. Not only the licensing cost is affordable, the license also includes the entire TGE source code. The TGE is a commercial grade game engine which has been successfully used in bridging the gap among multiple industry sectors (Shiratuddin & Fletcher, 2006).

During the investigation stage and prior to the development of the VDRS, in order to understand how the TGE works, the authors developed Figure 2 which shows the components and structure of it. Although documentation on the TGE was plentiful and can be found on GarageGames' website, most of the documents were outdated and did not apply directly to the development of the VDRS. In the investigation, the authors found that some of the TGE's documentation were redundant and no longer applies to current release of the TGE. This caused difficulty in understanding and finding the specific codes or functions. Some of the TGE codes spanned across multiple files in different folders therefore making the development process challenging. However, the TGE online community has provided assistance to the author. Publications by Finney (2007) and Maurina (2006) provided more organized information on the TGE.
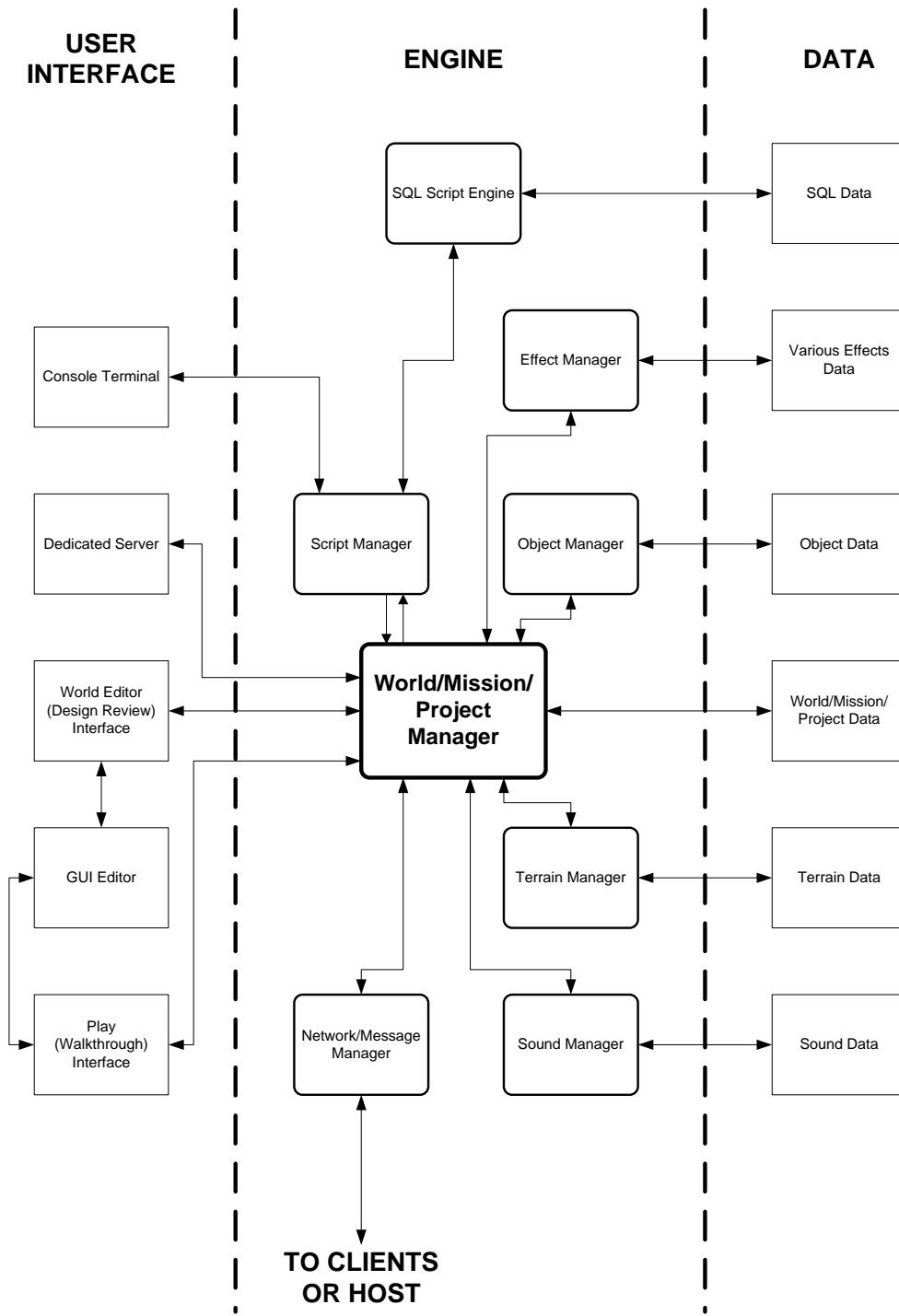
FIG 2: The components and structure of the TGE applied to the prototype application (Shiratuddin, 2009)

Referring to Figure 2, the 3 main components of the TGE are:

1) The **User interface** (UI) – this component consists of several sub-components i.e. the **Console Terminal**, **Dedicated Server**, the **World Editor**, the **GUI Editor** and the **Play** mode. The UI component comprised of both text and graphical-based UIs. The UI allows for interaction between the application and the end-user.

   The **Console Terminal** sub-component uses only text-based interaction. The console terminal can be of two types: (1) a text chat window and (2) the developer's debugging console terminal. While each type represents different functionality, both used text-based interaction with the end-user. The **Dedicated Server** sub-component allows for setting up and launching a dedicated standalone server through the **Console Terminal**. A dedicated server is usually setup to wait and listen, and accept incoming and outgoing connections from external clients. The server also synchronizes all the data between the clients and **World/Mission Manager**. The next UI sub-component is the **World Editor** or **Design Review** interface. The **World Editor** is where the visual scene assembling and design reviewing in the VE occurred. The **World Editor** employs a more graphical approach to many of the 3D object manipulation functionalities such as moving, deleting, rotating, scaling etc. Besides 3D object manipulation, the **World Editor** is closely linked to the **GUI Editor**. The **GUI Editor** also employs a visual approach in GUI design. It allows for the creation and modification of UIs for the design review prototype application such as adding new menu items, right-click menu, the login screen etc.

   The **Play** sub-component allows for real-time viewing of the VE. In the **Play** mode, only real-time walking-through the VE is allowed. No object or information manipulation exist is this mode. This mode is suitable for users who would like to only view the 3D model with no intention of making changes or comments to the design.

2) The **Engine** – this is the main component or the heart of TGE. The **Engine** component consists of various managers. Each manager is responsible to handle, retrieve and process specific type of data. At the center of the **Engine** is the **World/Mission Manager**. The **World/Mission Manager** manages all the interaction among other managers before passing the final required data to the end user.

   Examples of **Manager** are: the **SQL Script Engine Manager** that Manages information storage and retrieval; the **Effect Manager** that manages visual effects such as displaying the skybox, textures, lightings, highlights around an object when selected etc.; the **Object Manager** that manages all the objects in the VE which includes 3D objects, 2D GUI, avatars etc. It also manages properties such as texture mapping, geometry size, GUI size and color etc.; and etc.

3) **Data** – various data is stored and retrieved in different data elements. The required data for the VDRS are SQL data, various effects data, object data (textual and 3D object), world mission data, terrain data and sound data. E.g. Various **Effects Data** stores and retrieve effects such as glow, display of bounding boxes, lightings, shadows etc. **The SQL** data element stores and allows for retrieval of objects information and design review information.

3.3   Visualization of Design Review Information in a Virtual Environment (based on the principles of GUI Design)

What a user see in a VE is important so that it there will be a connection between the user and the VE. To consider a system as a VE, Burdea and Coiffet (2003) suggest that it has to have the *immersion, interaction* and *imagination* characteristics, while to Sheridan (1992) and (Slater *et al.*, 1998), *presence* is also a vital characteristic. These fours characteristics can only be experienced with a good interface design of the VE system. This interface is known as the *user interface* or *human interface* and in a VE system, many of the user interface will be visual or graphical, hence the word Graphical User Interface (GUI).

Mandel (1997) accounts three areas (also known as the Mandel's Golden Rules of Interface Design) of user interface design principles.

- Firstly, placing users in control of the interface. The user is free to go where they want to go and how they want to get there. E.g. user can work on a spreadsheet, switch to a web-browser to cross-reference information on the internet, and at the same time listen to audio music.
- Secondly, reduce users' memory load. The computer can assist the user to remember and carry out tasks that are repetitive. E.g. defaults such as, undo, redo, copy, and the use of icons.
- Thirdly, make the user interface consistent. Consistency is a key facet of a usable interface where users reuse their knowledge and what they learn to a new program. This can only happen if the programs are similar to what they have already used.

An important principle of GUI designs is to provide users with the tools and applications to do a task. GUI designs should also be concern with the ability for users to directly manipulate objects and information to do the task on-screen or the interface presented to them (Mandel 1997; Galitz 1997). The first thing to consider in presenting the information to the users is the amount of information the GUI should present at any one time. Helander (1988) highlighted that information clarity and readability is improved if information on-screen is showed in a less crowded fashion. The design of the interface should display only the information the user needs to perform his/her task or operation at hand. Simple window and icon designs, with unnecessary details results to less amount of time needed to complete tasks (Benbasat & Todd, 1993).

Secondly, the proper groupings of display of information. These grouping will lead to the information's readability and the relationships among the information that can easily be recognized (Helander, 1988). One way of grouping is to classify tasks within icons. Icons are usually small and do not consume too much screen space (Sears, 1993). Icons are also fast, easily recognizable and more visual than text (Benbasat & Todd 1993; Mandel, 1997). Novice user thus will find it easier to learn a system.

Thirdly, the sequencing of information on display. The layout of information should be in a manner that it is easy for users to navigate to find the information he/she needs to perform the tasks at hand. For example, users mostly expect the top of screen will always contain the headings for the pull-down menus, and scrollbars are in either side of the screen. GUI designers should also consider more important information to be placed at prominent locations. The goal of GUI is thus to allow users just to use the application on the computer and to concentrate on primary cognitive tasks, not to be concerned with the user interface. If attention is devoted to the interface, this will interfere with the user's main tasks (Benbasat & Todd 1993; Mandel 1997; Galitz 1997).

4. DEVELOPMENT OF THE VDRS USING THE TGE

Two major phases that were undertaken during the development of the VDRS; 1) the creation of the assets i.e. the 3BR house model and its matching textures, and the assembly of the VE, 2) the modification of existing functions or introducing new functionality through programming to support design review in a VE. In streamlining the development process, only certain tasks were undertaken at any particular time. These tasks include the preparation of the 3D model, designing and programming the GUI, improving 3D object manipulation functionality, introducing real-time collaborative design review across the network, and embedding database support for information processing. Through a series of phone and face-to-face interviews and meetings with local architects, inputs and feedbacks were gathered on the functionality, and the look-and-feel (focusing more on the GUI design and its functionality) of the VDRS. Figure 3 shows a breakdown of the development process and, the interaction between the authors and the architects.
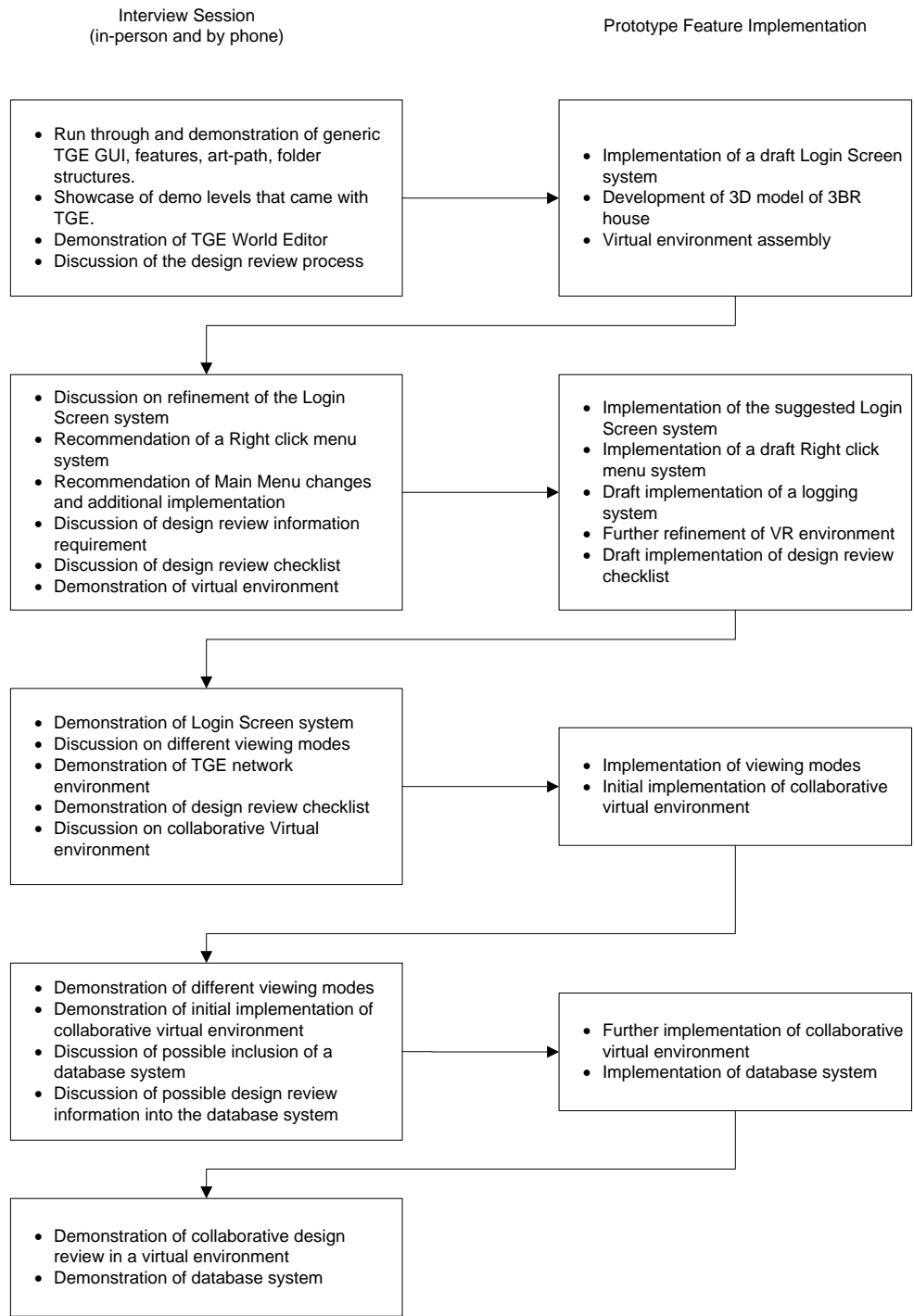
Interview Session (in-person and by phone)	Prototype Feature Implementation
• Run through and demonstration of generic TGE GUI, features, art-path, folder structures. • Showcase of demo levels that came with TGE. • Demonstration of TGE World Editor • Discussion of the design review process	• Implementation of a draft Login Screen system • Development of 3D model of 3BR house • Virtual environment assembly
• Discussion on refinement of the Login Screen system • Recommendation of a Right click menu system • Recommendation of Main Menu changes and additional implementation • Discussion of design review information requirement • Discussion of design review checklist • Demonstration of virtual environment	• Implementation of the suggested Login Screen system • Implementation of a draft Right click menu system • Draft implementation of a logging system • Further refinement of VR environment • Draft implementation of design review checklist
• Demonstration of Login Screen system • Discussion on different viewing modes • Demonstration of TGE network environment • Demonstration of design review checklist • Discussion on collaborative Virtual environment	• Implementation of viewing modes • Initial implementation of collaborative virtual environment
• Demonstration of different viewing modes • Demonstration of initial implementation of collaborative virtual environment • Discussion of possible inclusion of a database system • Discussion of possible design review information into the database system	• Further implementation of collaborative virtual environment • Implementation of database system
• Demonstration of collaborative design review in a virtual environment • Demonstration of database system	

FIG 3: Interviews and development of the VDRS

4.1 Virtual Environment Scene Assembling: Creating the 3D Model and Textures of the 3 Bedroom House

In this development phase, three main steps were involved. Step 1 was modeling of the 3BR house and preparation of the matching textures for all the 3D objects that made up the house. Step 2 was the placement of all the 3D elements and matching textures into specific TGE data folders. Step 3 involved the assembling of the VE using the TGE World Editor. A summary of the steps is shown in Figure 4.
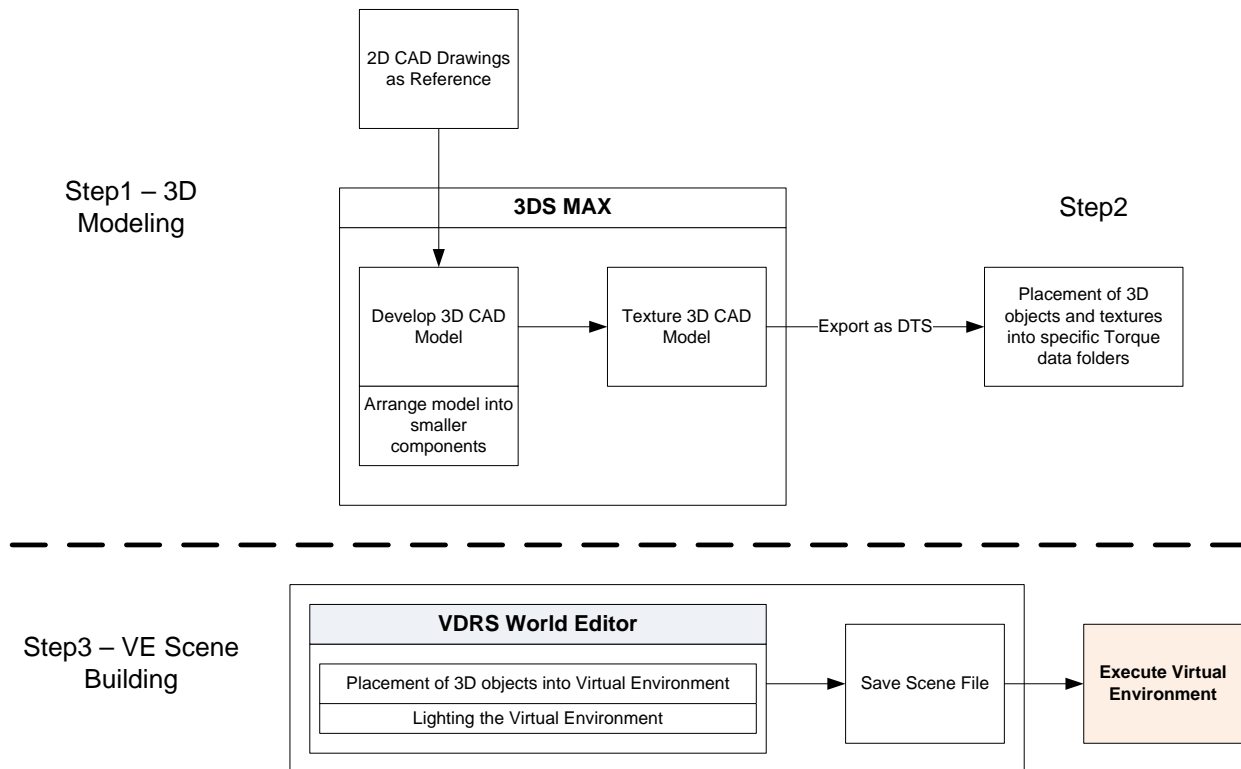

FIG 4: The workflow from 3D CAD into the VDRS

Step 1: Developing the 3D Model of the 3BR House

Based on a 2D CAD floor plan drawing, the 3BR house 3D model was created using Autodesk 3DS Max 9.0 3D modeling software. The house had to be divided into several sections. For example, similar 3D objects are grouped into distinguishable elements such as exterior walls, doors, windows etc. This step is important to avoid crashing the 3DS Max's TGE DTS exporter. Dividing the 3D model into smaller sections also helped to manage the 3D objects more easily. A 3D modeling layer management approach was used and this approach is similar in any 2D CAD design software where different types of elements are placed in different distinguishable layers.

Once the 3D modeling process was completed, various textures were applied accordingly to all 3D objects. The textures must either be in the JPG (JPEG – Joint Picture Expert Group) or PNG (Portable Network Graphics) image file format since these are the only texture file format supported by the TGE. Textures were created using a texture software creation application called MapZone developed by Allegorithmic (Figure 5).
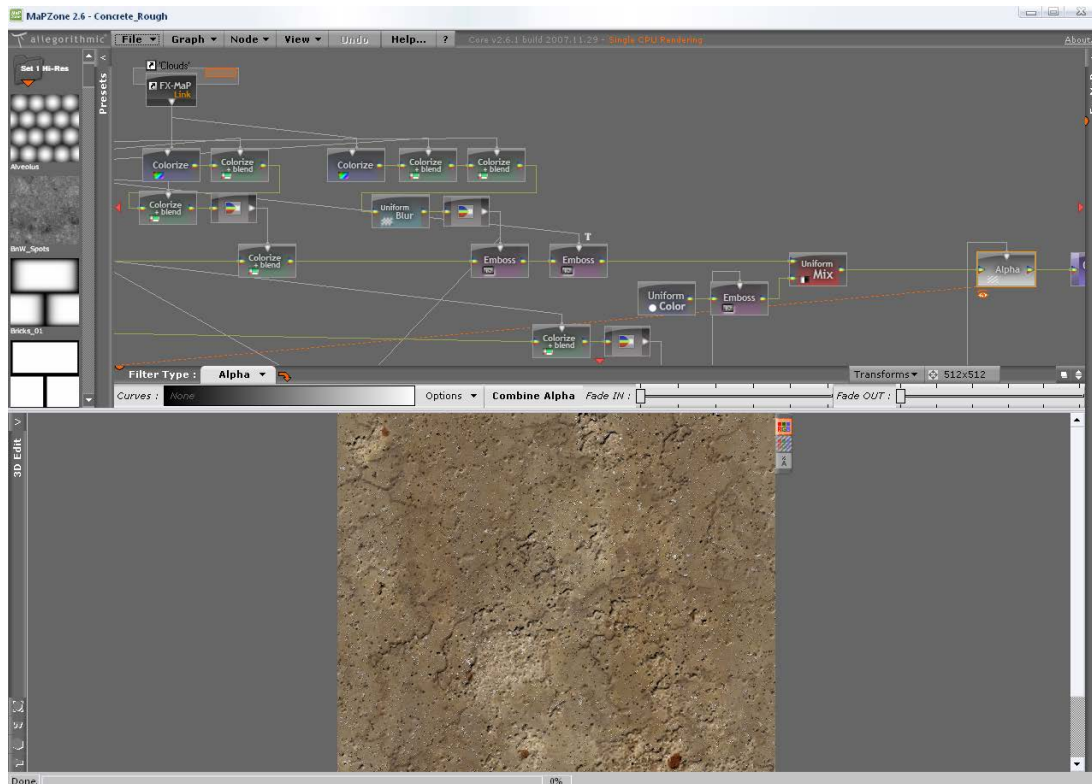
FIG 5: Screenshot of MapZone

Prior to exporting the 3D objects of the house into the TGE DTS file format, the 3DS Max to DTS exporter plug-in must be installed first, since by default 3DS Max does not come with the exporter. The DTS exporter was

downloaded from GarageGames' website. Once the exporter plug-in was installed, a utility icon [icon] was created on 3DS Max main menu bar for easier access. The DTS exporter plug-in made it easier to export 3D object into the TGE DTS file format. The 3BR house's 3D objects were then exported one group at a time into the DTS file format. Figure 6 shows rendered images of the 3BR in 3DS Max.
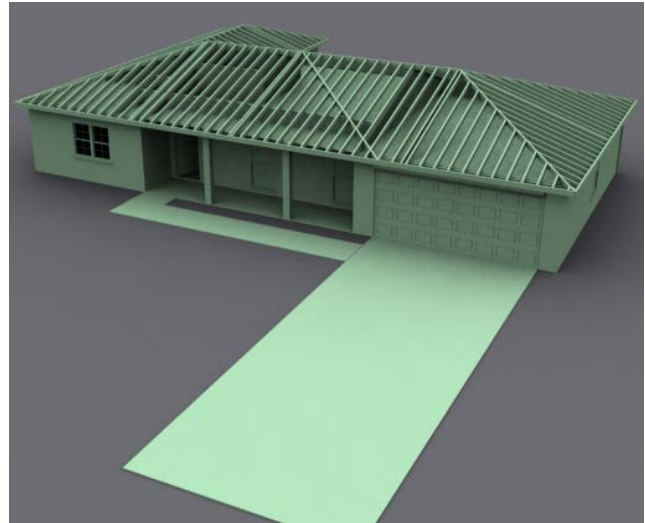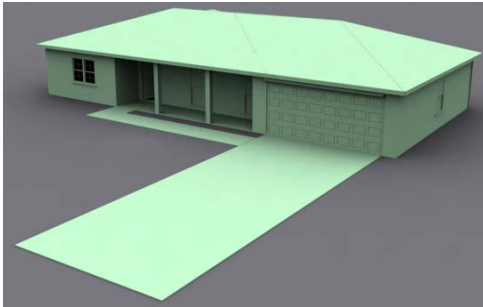
FIG 6: Rendered images of the 3BR house using in 3DS Max

Step 2: Copying the 3D objects and textures into the respective TGE folders

It is important to know where to place the 3D objects and their respective textures into the TGE folder structure. For the VDRS, the main folder structure is as shown below:

```
VDRS [root folder]
        |--common                [TGE common engine folder]
        |--creator               [TGE Editor functions folder]
        |-- projectOne           [specific project level folder]
                     |-- client
                     |-- data
                     |-- server
```

All the converted 3D objects were placed under the data folder. In the data folder, sub-folders were created for easier management and identification purposes such as *ceiling, door, ext_wall, floor, plumbing* etc. Each 3D object type was given its own folder, and the same data folder structure will appear in the VDRS.

Step 3: Virtual Environment Scene Assembling

As abovementioned, the data structure created in Step 2 was automatically recognized by the VDRS and was correctly displayed in the VDRS World Editor (Figure 7).
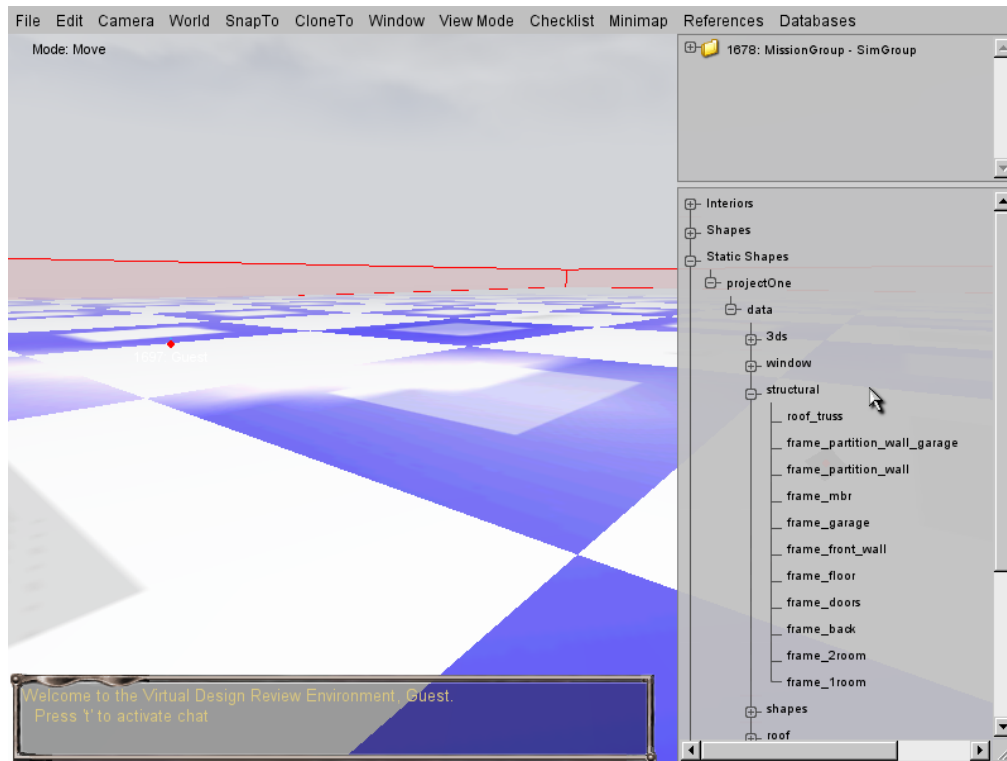
FIG 7: Data sub-folder structure in VDRS

The VDRS was launched by double-clicking the main application executable file. Since the VDRS scene assembly can only be done in the World Editor Creator mode (Figure 8), the F11 function key was used to switch to the World Editor mode. Next, by pressing the F4 function key, the World Editor Creator mode was invoked (or Main Menu → Window → World Editor Creator). In the World Editor Creator mode, 3D objects can be placed, arranged and aligned in the VDRS.
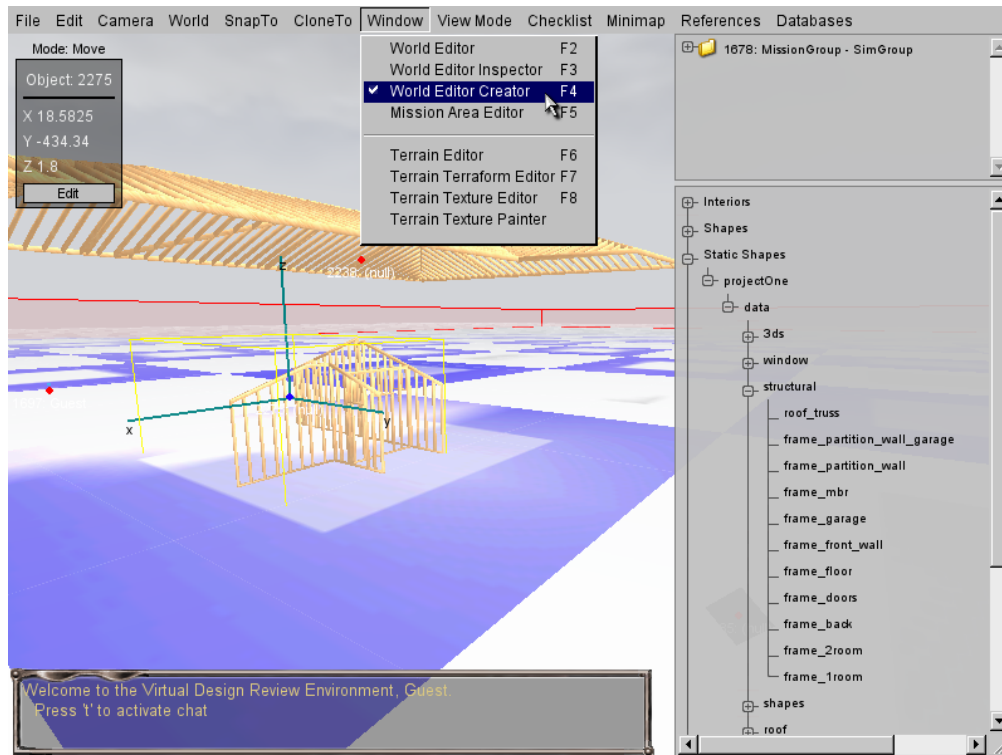
FIG 8: Launching the World Editor Creator for the scene building in the VDRS

One of the drawbacks of the TGE is it does not save the 3D objects coordinates from 3DS Max. This means that during the scene assembling process, all the 3D objects must be reassembled, placed and aligned properly. Besides programming the extra functionalities for the VDRS, reassembling the 3BR house model in the VDRS was one of the time-consuming processes. Different parts of the house were placed in by selecting it in the 3D objects library. The 3D object was manually moved and aligned to match with other corresponding 3D objects. This process continued until the whole 3BR house was reassembled (Figure 9).
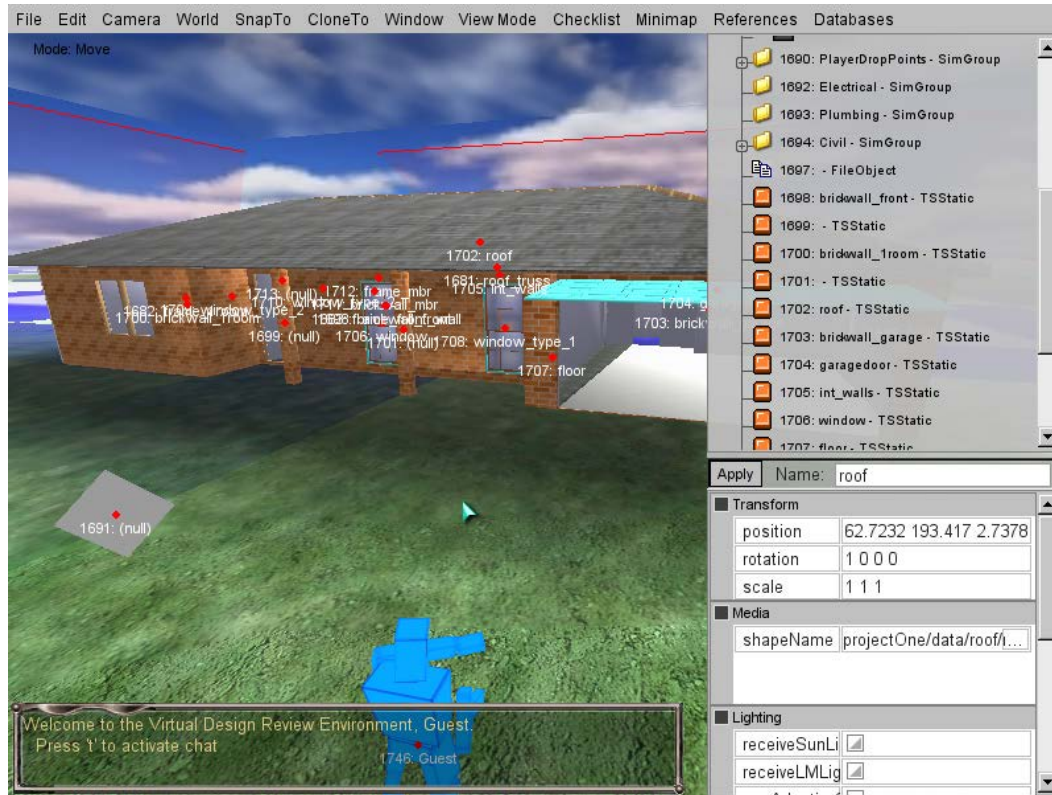
FIG 9: The completely assembled 3D model of the 3BR house

In easing out the manual realignment process and to ensure precision, a new feature was developed to allow parametric inputs through an on-screen GUI. User has the ability to key in the three-dimensional X-Y-Z transformation coordinates of an object. This feature was crucial as it allows for a more precise placement and alignment of 3D objects in the VDRS. The GUI was developed in such a way that three different transformation modes i.e. Move, Rotate and Scale, can be executed using the same on-screen GUI.

4.2 Programming the Features for the VDRS

Two types of programming were performed: (1) introducing totally new codes to extend the functionality of TGE to support the intended functionality of the VDRS, and (2) modifying the existing code structure to suit the VDRS's features.

4.2.1 Design of the Graphical User Interface (GUI) for the VDRS

The Graphical User Interface (GUI) is a combination of the graphics and the scripts that carries the visual manifestation of the VDRS. The GUI then accepts the user's control inputs such as mouse movement or keyboard presses. In the VDRS, part of the GUI elements includes: the player's Heads Up Display (HUD), the main start-up login screen window, the settings or option menus, the dialog boxes, the various in-world messaging systems such as text chat, right click menu, the hierarchical structure of the scene of the VE in the World Editor, and pull down menus. Many of the GUI scripts in the TGE share the same basic script layout and properties. Depending on the type

of GUI control, some may or may not require certain properties and therefore lines are added or omitted from the scripts themselves.

One of the useful development features of the TGE is the visual GUI designer. The F10 function key invoked the TGE GUI Designer (Figure 10). Using the TGE GUI Designer, GUI design for the VDRS was mainly developed visually. Many of the GUI controls were already provided with some basic embedded properties. Further customizations were performed by keying in the specific parameters in each given property box. Once a particular GUI design was developed, the TGE GUI Designer generated a custom GUI scripts for that particular GUI. Numerous GUI customizations were done to support the VDRS.
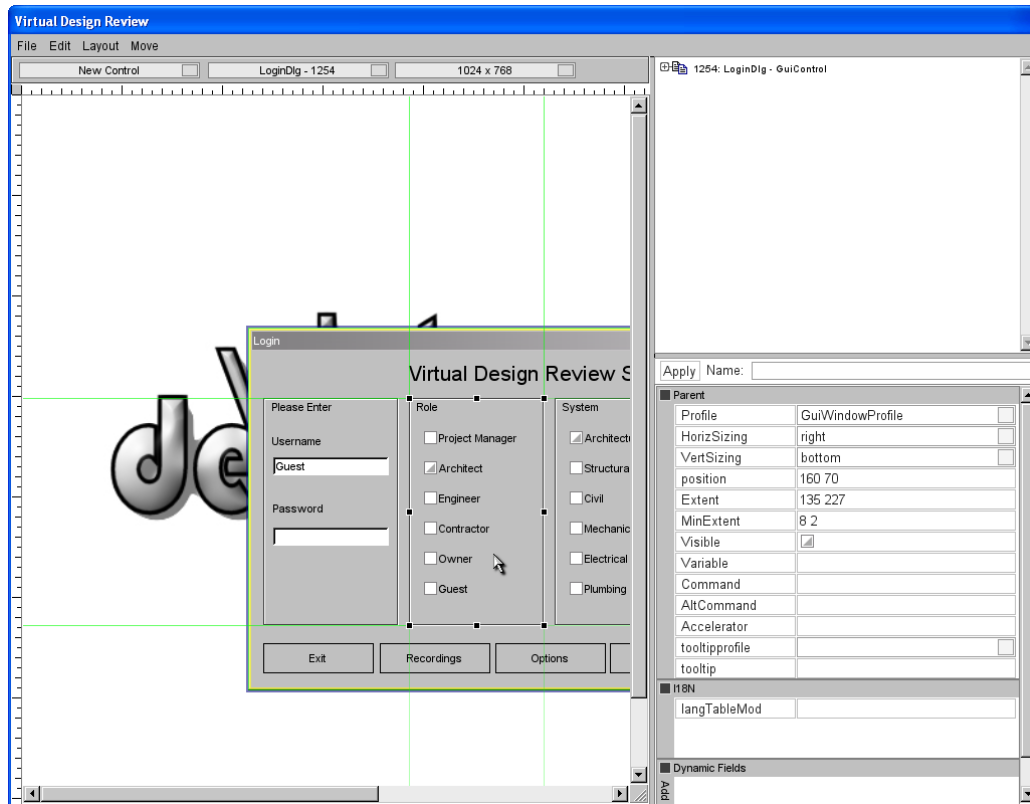

FIG 10: Example screenshot of the TGE GUI Designer window

The main login screen is the first screen design reviewers will see and it is the one of the most important features. Many of the parameters keyed in or selected by the design reviewer determined the level of information filtering. Since the default state of the TGE has no login system, new codes were developed for the VDRS. The VDRS login system also allows a design reviewer to host a design review session or join an existing session. It also allows a design reviewer to select a specific role and the building systems the reviewer wishes to review during a design review session.

Figure 11 shows the flow diagram of the login system of the VDRS. A user enters a username and password, then selects a role and, selects which building systems to review. The color of the Avatar will change based on the role selected. On the VDRS Main Login screen, the user can select whether to host a session or join as a client, edit the display options, view recording options, or exit from the VDRS.

If a user chooses to host a design review session, the user must select a project, check on the "Host Design Review Session" option, and then click on the "Launch Session" button. Any password from the VDRS Main Login screen will be attached to the session. Other users who wish to join in must use the correct password to log in. The "Host Design Review Session" option controls the availability/accessibility of the design review session to other users over a LAN or internet connection.

If a user chooses to join an existing design review session, the user must first locate a hosted session by hitting either the "Query LAN" or "Query Master" button. A search command will be issued to search for any existing session on the LAN and the internet respectively. To receive the most up-to-date information on a server that is hosting a design review session, the user can highlight it and use the "Refresh Server" button. Joining a hosted design review session is as simple as selecting it and hitting the "Join Server" button. As abovementioned, if a password exists on the server, it must be correctly entered on the VDRS Main Login screen.
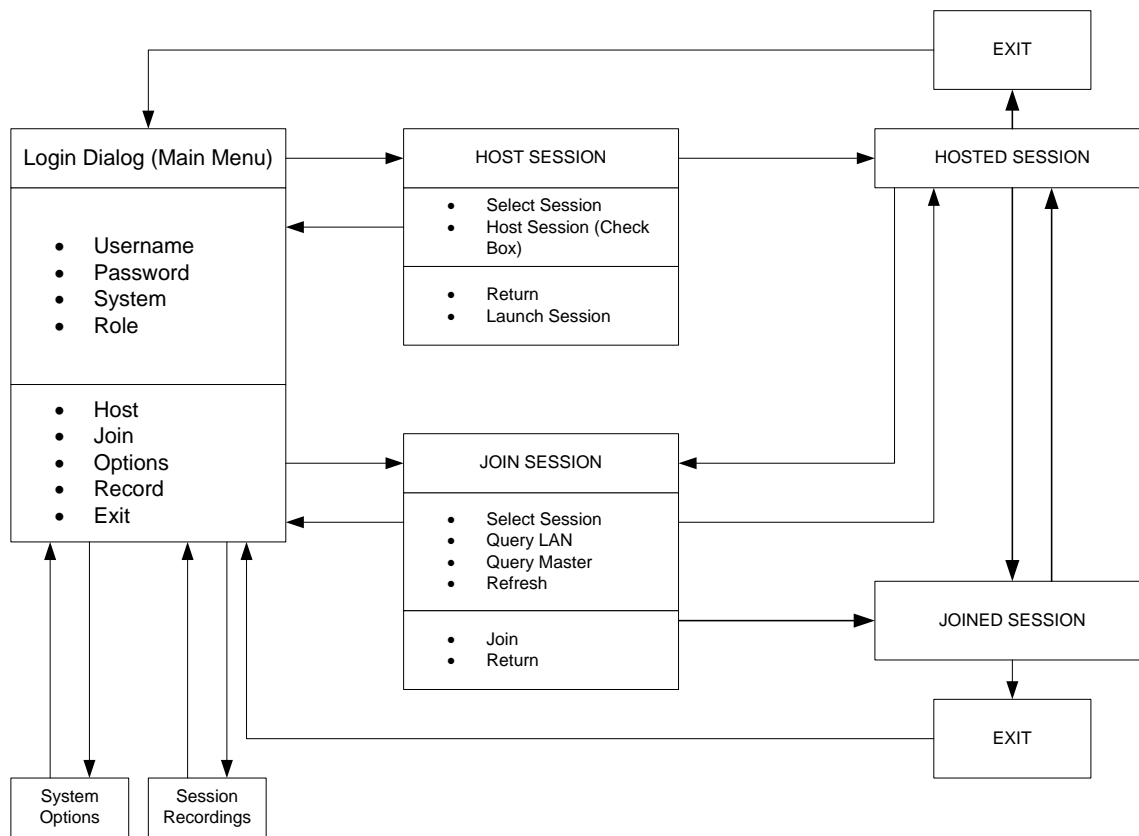

FIG 11: The flow diagram of the VDRS login system

4.2.2   3D Object Manipulation and the Right Click Menu System

From the discussion with local architects, it was recommended that implementing a right click menu system will be beneficial since the default implementation of TGE's World Editor was not intuitive enough. Many of the commands were either available only through the main menu, shortcut keys or console terminal. The architects were proficient 2D and 3D CAD users and thus preferred commands to be easily accessible through a right-click menu system.

However, for the VDRS, a cascading style (or a second level menu) right click menu system was not possible with the current available graphics functions in the TGE. The TGE graphics functions were limited to only 1-level menu system implementation. Adding this functionality may still be possible, but the TGE OpenGL functions will have to be rewritten.

Having only a 1-level menu system has limited the number of commands that can be made available on the right click menu. Once a draft right click menu layout was implemented, the authors consulted the architects again to clarify which commands were regarded important during a design review. The final implementation of the right click menu is as shown in Figure 12. As abovementioned, the right-click menu was another feature that was suggested by the architects and added to the VDRS. Besides the parametric input for the object transformation, seven 3D object manipulation features were developed to allow for easier assembly of the VDRS's virtual environment. These 3D object manipulation features can be access through the right-click menu.

FIG 12: The right-click menu

The function of each of the 3D object manipulation features introduced in the VDRS is shown in Table 1.

TABLE 1: Description of the commands available through the VDRS's right-click menu system

3D Object manipulation features	Function
Reset Rotation	Reset the rotation of the 3D object into its original state should there be any change in rotation made by the user.
Reset Scale	Reset the scale of the 3D object into its original state should there be any change in scale made by the user.
Hide Object	Hide a selected object or selection of objects.
Show All Objects	Show all previously hidden objects.
Delete Object	Delete an object.
Copy Object	Make a copy of the master object and allow for pasting that copy in the VE. The copied object does not inherit properties of the master object (which means that characteristics of the copied object can be change and it is independent of the characteristics of the master object).
Duplicate Object	Make an exact duplicate of the master object. Duplicated object will inherit any changes made to the master object. E.g. several doors were duplicated from a master door. Any changes made to master door will be reflected onto duplicated doors. This feature allows for mass property change for similar 3D objects.

When right clicking occurred, a collision test will be performed by the TGE. If an object besides the terrain, sky or water has been clicked, the selection will automatically be adjusted to include only that one particular object. A menu will then appear starting from the center of the screen, which is the position of the mouse and then is automatically transferred to the Object Manager right after a right click is performed. The menu contains actions to be performed on the single selected object as well as buttons to change the current state of the 3D transformation mode.

4.2.3   Viewing Modes

In supporting the nature of both local and collaborative design review in a VE, various viewing modes were developed. These included the FPV (First-Person-View), TPV (Third-Person-View), BEV (Bird-Eye View), OV (Orbital View) and BSV (Behind-Shoulder View). A FPV allows design reviewer to view objects in the VE from the avatar's eye. A TPV allows design reviewer to view objects in the VE from a camera view just above and behind the avatar. A BEV takes the design reviewer to view from a distance above looking down at the VE. An OV allows design reviewers to circularly view the VE as the center object.

BSV is a new viewing mode introduced by the authors to accommodate collaborative design review among design reviewers in a VE. A BSV allows one design reviewer to quickly switch the viewing FOV (field of view) to the location of another design reviewer, and have a view just behind that design reviewer's shoulder. This viewing mode allows for a design reviewer to view what another design reviewer is performing in the VE. The concept of BSV was originally thought by the authors since text-chatting and to manually travel to the location of another design reviewer was really cumbersome. The implementation was demonstrated to the architects and it was agreed that BSV was a

good viewing option especially during a collaborative design review session. The Tab key was programmed to accommodate the BSV feature.

4.2.4   Collaborative Design Review

The VDRS allows for real-time 3D object manipulation not only on a local PC, but also across the network with PCs connected either through the LAN or the internet. In a collaborative design review session, all forms of data are tracked on all client PCs that are connected to a Hosting Server. A client can now join a host PC over a LAN or internet connection, and perform 3D manipulation tasks in an almost identical fashion to that of the host. 3D manipulation tasks include moving, rotating and scaling objects, creating objects, deleting objects, and copying and pasting objects. These tasks can be executed simultaneously in real-time on the host as well as on other client PCs.

Numerous changes were made to the TGE codes so that the client PC is able to enter the World Editor mode (or Review mode) and load the scene just like the host is able to. Once the client is in the World Editor mode, custom codes had to be written to translate each command the client performs on the VE, back to the server. Each manipulation in the VE results in a server command message being sent to the host with details on which objects are being manipulated and the new condition of the object. The server receives these server command messages and executes them in the order they are received. To avoid messaging overloads which can affect the real-time performance of the VE, the number of messages the client sends to the host per second are adjusted accordingly. Once the host has executed an update message, the object in reference is updated back out to the clients just like it would have been with a traditional scene in the TGE. The host can now adjust how many times within a second these updates are sent out to the clients, to help preserve the network bandwidth if there are too many clients reviewing the design simultaneously in real-time. Figure 13 shows a generic representation of the inner-workings of the messaging system of the VDRS.
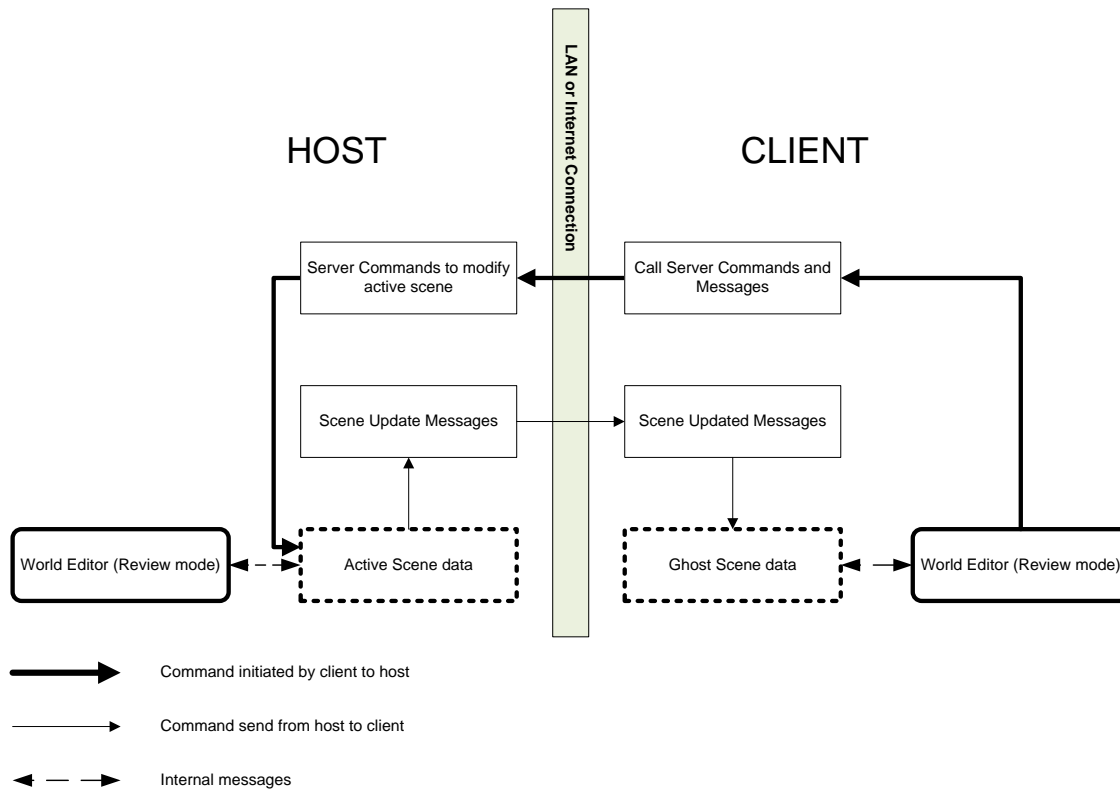
FIG 13: An overview of the VDRS messaging system

4.2.5   Database Support for the VDRS

As mentioned earlier, the TGE is a low cost and flexible 3D game engine. However, there are still limitations when it comes to developing real-world and a more serious application. One of the proposed components for the VDRS is the processing of design review information processing that includes data and information storage and retrieval. In supporting the processing of design review information, the authors decided to use an open-source database software called SQLite. SQLite is different from SQL in that it offers no built-in server commands. This makes SQLite ideal for software which already has a networking infrastructure, like the TGE and hence the VDRS. SQLite is specifically designed to be added to an existing implementation of a software that will in turn enables that software to have a self-contained, persistent database system which requires no configuration. For example, the easiest way to create persistent editable data with the TGE would be to use a text file format (such as comma-separated values: CSV and tab delimited text: TDT). CSV and TDT text file format can be opened and edited using most modern spreadsheet software such as Microsoft Excel. However, using such file format has many problems. First, the data would be readable and editable using any text-capable reading software such a Notepad or WordPad. Second, managing the data would become increasingly time-consuming and difficult as the data becomes larger. It may be easier to read in, processed and then sent out a few lines of information but for larger tasks, such as storing design review information, a proper database system is needed. In the VDRS, a rule-based approach was implemented using the CSV text file format, as it is easier to change or add items in the design review checklist, and there was no large data storage requirement. Whoever is responsible for the overall project's design would be able to modify design review checklists that can later be used by design reviewers.

Once SQLite was integrated into the VDRS, it creates a quick reference, encrypted database system which allows for many commands such as creating, deleting, querying, editing, and reorganizing the data. Some of the uses of SQLite in the VDRS are logging the design reviewer's login information; store, retrieve and display design review comments and building codes; tracking object movements and embedded information such as dimensions, material; and cost of the objects.

Figure 14 shows how information is bi-directionally passed between the user and the SQLite databases. Information is processed based on the user's request. The information is passed from the user through the Torque scripting code, and then through the SQLite code which is embedded in the TGE. From the TGE, the information is stored into the databases as persistent data. A *persistent data* is a term used to describe any data that will need to be written (stored) to a disk (storage device) and later can be retrieved and read back for further processing. In the VDRS, design review information is being passed between the user and the system.
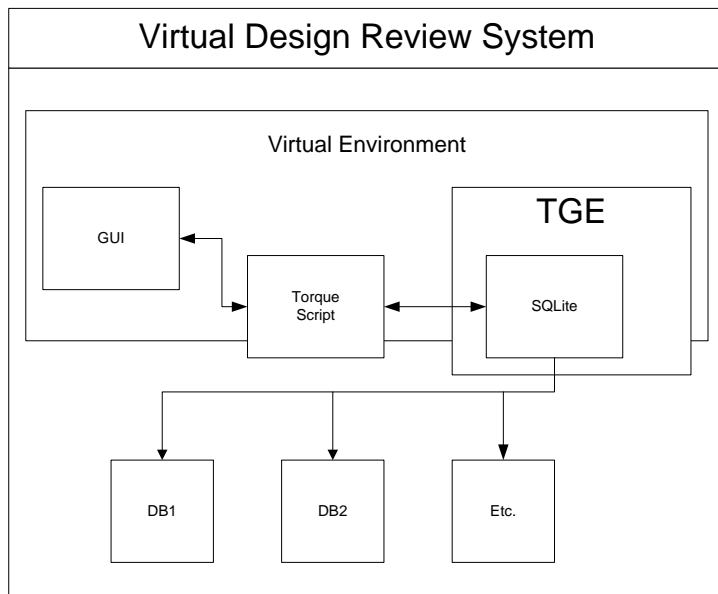

FIG 14: Data flow diagram for the information passing through one component to the other

The users often input data through the VDRS's GUI system. The GUI implementations were in the forms of editable text boxes or multi-line editable text boxes. This information is then passed to the SQLite engine code in a string format using scripting code. The script handles all the details of each command so a user does not have to understand all the jargon needed to communicate with the SQLite engine. This feature makes the GUI design to be as simple as point and click, provided that the GUI is set up to execute such commands. Once the GUI system executes the string commands, the string data is interpreted by the SQLite engine and then acts on the designated database according to the interpretation it derives from the string data it received.

The example shown below describes how a design reviewer would enter a set of specifications to an existing 3D object that does not have one. In the VDRS, when a 3D object is selected, a query is automatically run on the object. The query is based on the object model's name (object-centric mode). The query is run to determine if any specifications have been previously assigned to the object. If the specifications do exist, a pop-up window will appear to allow the design reviewer to click and view the specifications information. If no specifications information has been defined, the same popup window will appear. The design reviewer can click on the "Create Specifications' button. An empty "Edit Specifications" dialog box will appear and the design reviewer can enter the appropriate

information for the 3D object. Once completed, the design reviewer can click the "Save and Return" button (Figure 15).
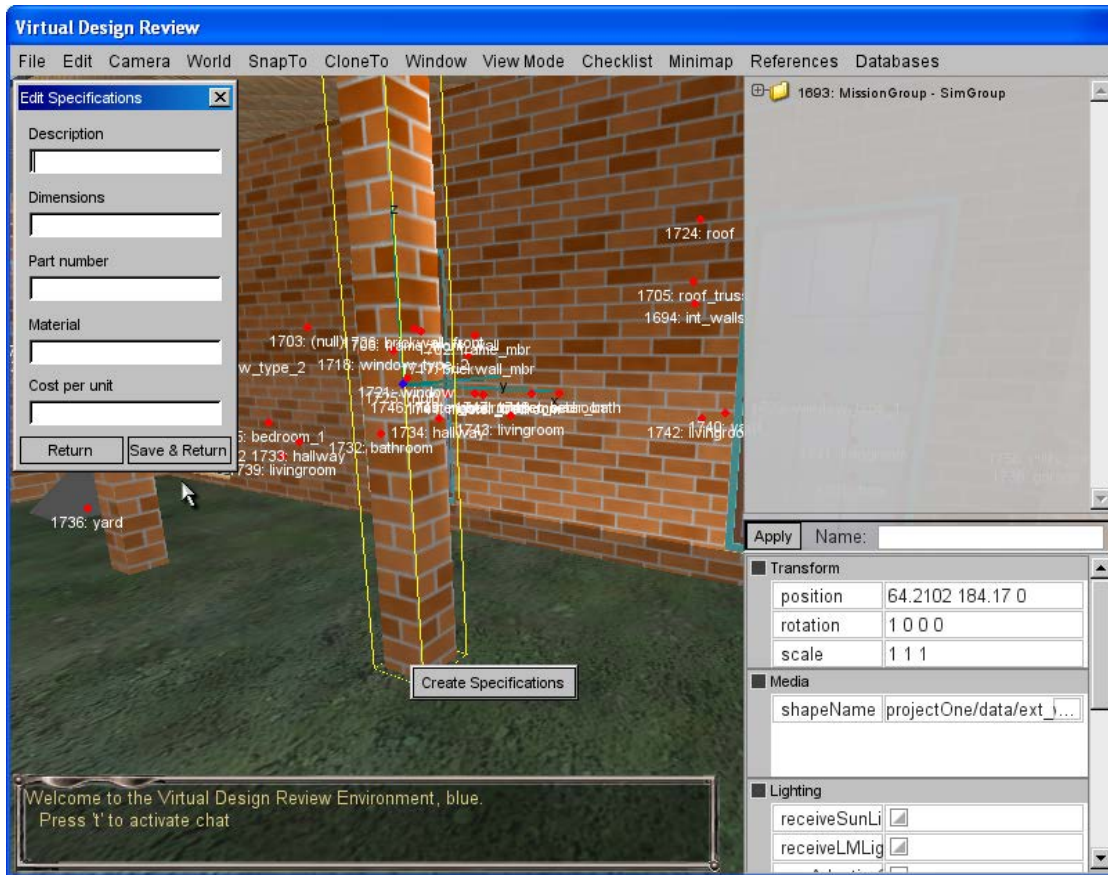

FIG 15: The Edit Specification window to allow for addition of new information

This action assigns the data to the SQLite database as defined by the SpecsEditDlg::onSave function. Upon assigning the specification information to a 3D object, the dialog box which displays the specifications information is automatically opened and displays the data of the object. The SpecsMenu::onWake function then queries and returns the data from the SQLite database. It then displays the information to the user in the specifications dialog box. All object specifications information is stored in the SPECS SQLite database file. Objects are sorted first by model name (description) within the SPECS file. All other data such as cost, dimensions, material etc. are created and stored as secondary fields. The specification information can be edited and updated at any time.

5. CONCLUSION

In this paper, we have described the components, the structure and the development of the VDRS using the TGE. In liaising with the intended purposes of the VDRS, only suitable and usable components of the TGE were used and incorporated into the VDRS. Two major development phases took place; development of the VE, and the modification of existing or the introduction of new functionalities through programming (such as the implementation of real-time 3D object manipulation, real-time collaboration, networking, GUI, information processing and

visualization). The VDRS was developed in several stages, and each time a milestone was achieved, the authors consulted with local architects for their expert opinion.

All the design review related features developed are new, and none of them exist in the stock TGE (with the exception of FPS, TPS, Orbital and Bird's Eye view mode). Some of the specific design review features implemented in the VDRS are shown in Table 2.

TABLE 2: Features developed for the VDRS

	Feature	Description
1	Database support	Dynamic databases created using SQLite are fully supported in the VDRS. Some of the databases implemented are logs and object information. A lessons-learned database was also created.
2	Real-time collaboration across network	Real-time collaboration across the network supports 3D object manipulation, text-chat and various viewing mode.
3	View modes & Navigation	First-Person, Third-Person, Orbital, Birds-Eye, Behind-Shoulder were implemented to support collaborative design review. The Behind-Shoulder view is a new concept developed by the authors. Navigations in the VE were closely tied to the viewing modes. Flythrough and Walkthrough in the VE were the two navigation modes implemented in the VDRS.
5	3D Object Manipulation	Objects can be placed, moved, scaled, rotated, and deleted in the VE.
6	Parametric input	Parametric input was implemented as an option to object-manipulation in the VE. Parametric input helps precise object movement, scaling and rotation.
7	GUI	Various GUIs including login screen and right-click menu were developed to support various design review tasks in the VE.
8	3D Object database and organization	A more organized 3D object directory file structure was introduced. The default hierarchical-tree support present in the TGE was improved. 3D object can be hide or shown through right-click.

Based on the successful development and features of the VDRS, there is thus a need for design review related information to be centralized, easily accessible and processed to assist design reviewers during design review. Successful and automated design review system will reduce the number of reviewers and the time necessary to complete the review phases, thus, shorten the project life-cycle, earlier occupancy can be accomplished which ultimately translates into significant total project savings to the owner. There is also a need to represent designs not just in 2D but 3D, where designers, reviewers, and other project team members can collaborate, view and review the same 3D model in real-time within the same 3D virtual space. This form of collaboration ensures that everyone impacted by the design has early access to the design and design review information and has the ability to influence the final design at an earlier stage of the design process. A means for real-time collaboration within the same 3D virtual space is therefore needed to be included in the design review process. Utilizing a 3D game engine based VE

technology not only can improve the 3D representation of the design that in turn will assist and provide a common language for the project stakeholders, it will also help developers of VEs to utilize current state of the art technologies that were at one point in time, available only for games development purposes. The underlying strengths of 3D game engines such as real-time 3D rendering, real-time walkthrough, interactivity, multi-participatory, lighting, collision detection etc, can be harnessed into the development of real-world applications such as the Virtual Design Review System (VDRS).

6. REFERENCES

Benbasat, I., and Todd, P. (1993). An experimental investigation of interface design alternatives: icon vs. text and direct manipulation vs. menus. *International Journal of Man-Machine Studies, 38,* 369-402.

Burdea, G., & Coiffet, P. (2003). *Virtual Reality Technology*. Hoboken, New Jersey: John Wiley and Sons.

Campbell, D.A., & Wells, M. (1994*). A critique of virtual reality in the architectural design process*. From University of Washington, Human Interface Technology Laboratory Website: http://www.hitl.washington.edu/publications/r-94-3/

East, W. (1998). Web-Enabled Design Review and Lessons Learned. Champaign, IL: U.S. Army Corps of Engineers Construction Engineering Research Laboratories. (CERL Report No. A 396443).

East, W., Kirby, J.G., & Perez, G. (2004, April). Improved design review through web collaboration. *Journal of Management in Engineering*, 51-53

East, W., Roessler, T.L., & Lustig, M. (1995a). Improving the design-review process: the Reviewer's Assistant. *Journal of Computing in Civil Engineering, Oct 1995,* 229-235.

East, W.E., Roessler, T.L., Lustig, M.D., & Fu, M. (1995). The Reviewer's Assistant System: System Design Analysis and Description. Champaign, IL: U.S. Army Corp of Engineers Construction Engineering Research Laboratories. (CERL Report No. FF-95/09/ADA294605).

Finney, K.C. (2007). *3D game programming all in one (2nd ed.).* Boston, MA: Thomson

Fu, Michael C., & East, E. W. (1999). The Virtual Design Review. *Computer-Aided Civil and Infrastructure Engineering, 14*, 25-35.

Galitz, W.O. (1997). *The essential guide to user interface design: an introduction to GUI design principles and techniques.* New York, NY: John Wiley & Sons.

Garagegames.com

Helander, M. (1988). *Handbook of Human-Computer Interaction.* New York: North-Holland Publishing.

Ichida, T. (1996). *Product design review: a methodology for error-free product development*. Portland OR: Productivity Press.

Mandel, T (1997). *The elements of user interface design*. New York: John Wiley & Sons, Inc.

Maurina, E.F. (2006). *The game programmer's guide to Torque: under the hood of the Torque Game Engine.* Wellesley, MA: A K Peters Ltd.

Mays, P. (1998, October) Making virtual reality real. *Architecture*, 87(10), 162.

Miliano, V. (1999). *Unrealty: application of a 3D Game Engine to enhance the design, visualization and presentation of commercial real estate*. Retrieved March 20, 2008 from http://www.vsmm.org/vsmm99/

Mullen (1998) Mullen, M. (1998). *CGW's New 3D GameGauge. GameSpot*. Retrieved September 4, 2000 from http://www.zdnet.com/gamespot/stories/news/0,10870,2463249,00.html

Nigro, W.T., & Nigro, W.M. (1992). *Redicheck Interdisciplinary Coordination (3rd ed.).* Peachtree City, GA: The REDICHECK Firm.

Pugh, S. (1990). *Total design - Integrated methods for successful product engineering*. New York: Addison Wesley Publishing.

Sears, A. (1993). Layout appropriateness: a metric for evaluating user interface widget layout. *IEEE Transactions on Software Engineering, 19(7),* 707-720.

Sheridan, T. (1992). Musing on telepresence and virtual presence. Presence: *Teleoperators and Virtual Environments, 1(1),* 120–125.

Shiratuddin, M.F. (2009). A Framework for Design Review in a Virtual Environment: Using Context Aware Information Processing, ISBN-10: 363917285X, ISBN-13: 978-3639172850, Publisher: VDM Verlag, Pub. Date: August, 2009.

Slater, M., Steed, A., McCarthy, J., & Maringelli, F. (1998). The influence of body movement on subjective presence in VEs. *Human Factors Journal, 40(3),* 469-477.

Spillinger, Ralph S. (2000). *Adding Value to the Facility Acquisition Process: Best Practices for Reviewing Facility Designs.* Retrieved March 21, 2008 from the National Academies Press Website: http://www.nap.edu/catalog.php?record_id=9769#toc

Staub-French, S., & Fischer, M. (2001). Industrial Case Study of Electronic Design, Cost, and Schedule Integration. CIFE Technical Report #122, Stanford

Vince, J. (1995). *Virtual reality systems*. Addison Wesley Longman.