Development of a Collaborative Design Tool for Virtual Environment (CDT-VE) Utilizing a 3D Game Engine

Mohd Fairuz Shiratuddin
mohd.shiratuddin@usm.edu

Jason Breland
jason.s.breland@usm.edu

School of Construction
The University of Southern Mississippi, Hattiesburg, MS 39406, USA

Abstract

Collaboration in architectural design can be enhanced by using a virtual environment (VE). The visual aspects in a VE facilitates shared understanding across interdisciplinary groups, further enhanced 3D models, provide visualization support, and allows for the generation of alternative designs. The design team can collaboratively design, visualize, review, redesign and solve design problems and errors that may arise before actual construction commences on site. A Collaborative Design Tool for a Virtual Environment (CDT-VE) enhances the collaborative ability for designers. Designers can work on the same design within the same space, without interfering with one another, in a higher level of interaction. This paper describes the development process of a CDT-VE. The CDT-VE uses the Torque 3D Game Engine (TGE).

Keywords: collaborative, design, game engine, torque, virtual environment

1 Introduction

Early collaboration with the support of computer technology benefits designers and the consequent processes in a construction project that are often complex (Maher et al, 1996). As noted by Howard *et al* (1989) a construction project is a complex endeavor and its success is most likely only when different professionals collaborate especially from the early stages. During a study of computer mediated architectural design, Maher *et al* (1996) observed three forms of collaboration; *mutual collaboration* which involves designers equally working together on the same aspect of the task; *exclusive collaboration* is when designers working on separate aspects of the same problem with occasional time for consultation; and *dictatorial collaboration* when by appointment or naturally, there emerge a "designer in charge" who makes all the design decisions. This study also found that computer plays an active role in collaborative design because it provides designers with the visualization support for 3D models, assistance in generating alternative designs, and a platform for design critiques.

A virtual environment (VE) can be crafted to support collaborative work (Bryson, 1996; Whyte *et al*, 1999; Schuckmann *et al*, 1999; Theoktisto & Fairen, 2005; Liston *et al*, 2000). Users of VEs benefit from the visual that is presented to them because it facilitates a shared understanding. With shared understanding, collaboration is realizable and extendable across interdisciplinary groups. A VE can further enhanced 3D models, provide visualization support, generate alternative designs, and ultimately provides a platform for collaboration for a design team. The design team can collaboratively design, visualize, review, redesign and solve design problems and errors that may arise before construction commences on site.

We have developed a Collaborative Design Tool for a Virtual Environment (CDT-VE) that enhances collaborative ability for designers. Using this tool, designers have higher levels of interaction, where they can work on the same design within the same space, without interfering with one another. Our CDT-VE is very much complete and capable of standing on its own as a working model. In this paper we define CDT-VE as a VE that supports synchronized multi-participants design activities. A designer is able to co-exist within the same time and virtual space with another designer. They are able to see what others are doing in the VE. A CDT-VE that is meant for architectural design would be a perfect solution to enhance collaboration among designers.

In this paper, we describe the development process of a CDT-VE. The tool is developed to assist in a study (that will be conducted in Fall 2008 and Spring 2009) on how architecture students (as future designers) perform design collaboratively in a real-time VE. We will also study the suitable graphical user interfaces (GUIs) for design tasks in a real-time collaborative VE. There are still a few new features to be added to the tool. The results will reveal the benefits of using such tool in addition to and in comparison to other forms of traditional collaboration.

We use GarageGames' Torque 3D Game Engine (TGE) to develop the CDT-VE. We use the TGE not only because of its affordable licensing cost (USD $150), but the license also includes access to the entire source code of TGE. The TGE is a commercial grade 3D game engine which has been successful in bridging the gap among multiple industry sectors (Shiratuddin & Fletcher, 2006). Based on our experience with other 3D game engines such as Epic Games's Unreal engine and Valve's Half-Life engine, we conclude that without access to the entire engine's source code, further modifications of the core structure of the 3D game engine to support real-world applications development are impossible (Shiratuddin & Thabet, 2003a). Figure-1 shows the components and structure of the TGE that were used for the CDT-VE.

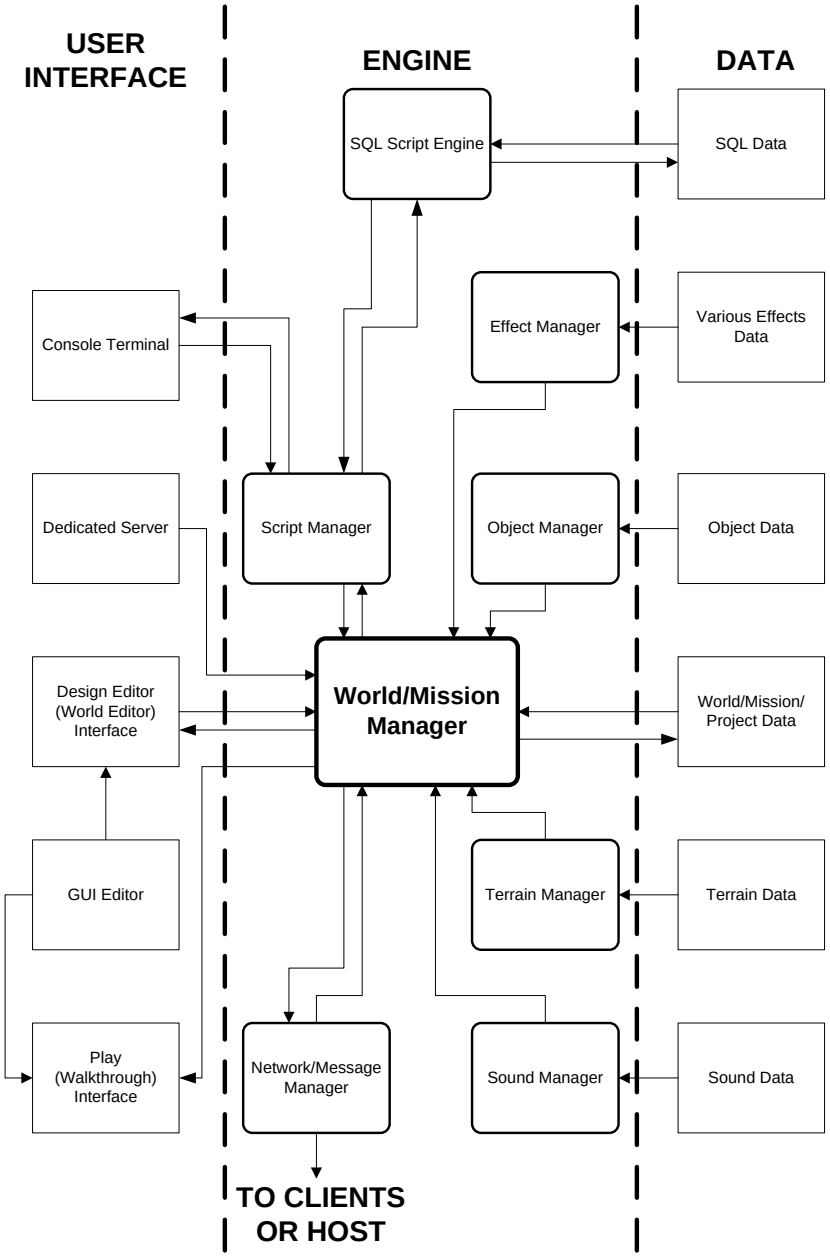

Figure-1 - The components and structure of the TGE that were used in the CDT-VE

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. In the CDT-VE, the console terminal is used for: (1) a text chat window and (2) the developer's debugging 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 using the **Console Terminal**. A dedicated server PC is usually setup to wait and listen, and accept (or reject) any incoming and outgoing connections from external client PCs. The server also synchronizes all the data between the client PC and the **World/Mission Manager**. The next UI sub-component is the **World Editor** or **Collaborative Design** interface. Collaborative design activities occur in the VE through the **World Editor**. The **World Editor** employs a more graphical oriented approach to 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** uses a visual approach for GUI design. The **GUI Editor** allows for the creation and modification of UIs for the CDT-VE such as adding and designing new menu items. Next, is the **Play** sub-component. It allows for real-time viewing of the VE. In the **Play** mode, only real-time walk-through the is allowed. No design functionalities exist is this mode. The **Play** mode is suitable for users such as the owner who would only like to view the 3D design with no intention of making changes to it.

2) The **Engine** is the main component of the TGE. It consists of various managers, and 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 between other managers before passing the final required data to the end user. Table-1 shows descriptions of the various managers that exist in the engine component.

Manager	Description
SQL Script Engine	Responsible for information processing such as store and retrieve.
Effect Manager	Covers the various visual effect elements such as displaying the skybox, textures, lightings, highlights around the object when selected etc.
Script Manager	Manages the execution of any scripts or functions required by other managers and then passes it to the World Manager and Console Terminal.
Object Manager	Responsible for managing all the objects presents in the VE. This includes 3D objects, 2D GUI, avatars etc., and their properties such as texture mapping, geometry size, GUI size and color etc. Any items present in the VE are treated as object.
World/Mission Manager	All data must go through the World Manager. The data is redistributed accordingly to other managers or to the end-user. The World Manager tracks the location of each object present in the VE.
Terrain Manager	Manages the rendering of the terrain. The terrain engine is part of the terrain manager and it is a sub-set of the engine.
Network/Message Manager	The Network Manager manages any data that requires updating across the network. Whenever real-time collaboration occurs, any changes made by any designers will be relayed to the hosting server and then retransmitted to other client PCs connected to the collaborative design session..
Sound Manager	Manages all sound related data. This includes sound cues in the VE, sound effect when the avatar is walking in the VE, etc.

Table-1 - Description of the engine's managers

3) Various data is stored and retrieved in different data elements. The required data for the CDT-VE are SQL data, various effects data, object data (textual and 3D object), world mission data, terrain data and sound data. Table-2 describes the various types of data stored in the data component.

Data	Description
SQL data	Stores information in various databases in the SQL format. Information stored here includes log files for each designers and information on the 3D objects.

Effects data	Stores visual effect related data such the skybox, textures, lightings, highlight color around the object when selected in the VE, etc.
Object data	Stores data of all the objects presents in the VE. These include 3D objects, 2D GUI, avatars etc., and their properties such as texture mapping, geometry size, GUI size and color, etc.
World/Mission/Project data	Stores the world/mission file so it can be retrieved later. The mission file contains the design data.
Terrain data	Stores the terrain data such as heightfield elevation map, the terrain geometry and terrain textures.
Sound data	Stores the sound data for the VE. These include sound effects such as footsteps, sound cues and background music.

Table-2 - Description of the types of data

2 Development of the Collaborative Design Tool for a Virtual Environment (CDT-VE)

The CDT-VE is a highly modified version of the TGE. Using C++ and C-like syntax scripting language, additional codes were written to provide the added functionalities specifically for design purposes. Microsoft Visual Studio 2005 was installed on an IBM-PC compatible desktop PC and was used as the programming IDE (integrated development environment). The development of the CDT-VE involves two major phases: 1) Development of the VE. We use a 3D model of a 3BR house in our planned experiment. Using pre-built components, as a design team working collaboratively, students will assemble the 3BR house in the VE, and 2) Programming of existing or the introduction of new functionalities such as the implementation of real-time 3D object manipulation, real-time collaboration, networking, GUI, information processing and visualization. These functionalities are important to ensure the collaborative design process in a VE is achievable.

In the development phase of the VE, three main steps were involved. Step 1 was the modeling of the 3BR house using Autodesk 3DS Max, and the preparation of the matching textures for all the 3D objects that made up the house. We divided the house into several sections and grouped similar 3D objects into distinguishable group e.g. exterior walls, doors, windows etc. This was an important step to avoid crashing the 3DS Max's TGE DTS exporter. 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 Allegorithmic MapZone software. The 3BR house's 3D objects were exported one group at a time into the TGE's DTS file format. Figure-2 shows rendered images of the 3BR in 3DS Max.

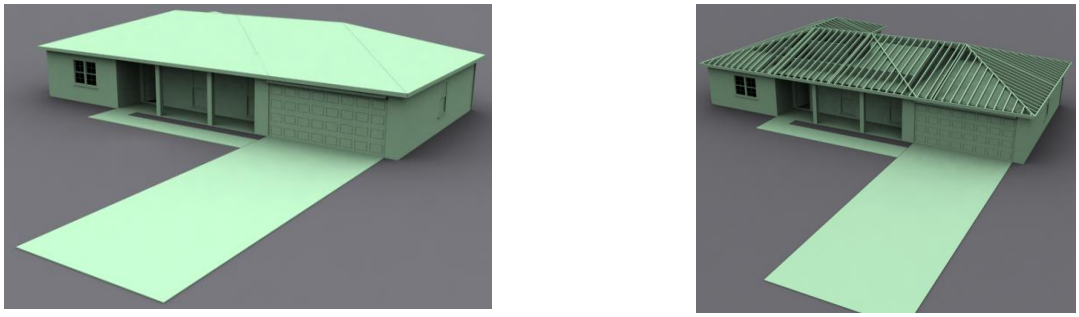

Figure-2 - Rendered images of the 3BR using the Light Tracer rendering engine in 3DS Max

Step 2 was the placement of all the 3D elements and matching textures into specific TGE data directories. Step 3 involved the assembling of the VE. The data structure created in Step 2 was automatically recognized by the TGE and was correctly displayed in the World Editor of the CDT-VE. In the CDT-VE, the VE scene assembling can only be done in the World Editor Creator mode. In the World Editor Creator mode, 3D components of the 3BR house were placed, arranged and aligned in the VE. We found that one of the drawbacks of TGE is it does not save the 3D objects coordinates from 3DS Max. This means that during the VE scene assembling process, all the 3D objects must be reassembled, placed and aligned properly. Besides programming the extra functionalities for the CDT-VE, reassembling the 3BR house model in the VE was one of the time-consuming processes. Different parts of the house

were placed in the VE by selecting each one of the house's components from the 3D objects library. The selected 3D object was manually moved and aligned to match with other corresponding 3D objects in the VE. This process continued until the whole house was reassembled (Figure-3).

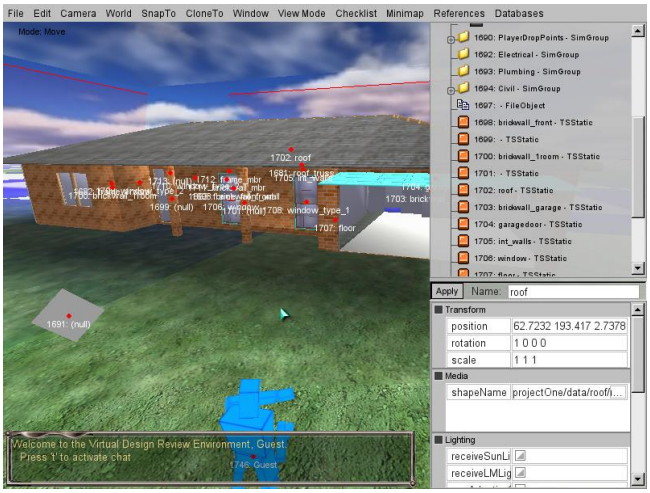

Figure-3 - The completely assembled 3D model of the 3BR house

In easing out the manual realignment process and to ensure precision, a new feature was programmed to allow parametric inputs through an on-screen GUI (Figure-4). With this feature, designers have the ability to key in the three-dimensional X-Y-Z transformation coordinates of a selected 3D object. We think this feature is important as it allows for a more precise placement and alignment of 3D objects in the VE. The GUI was designed and programmed in such a way that three different transformation modes i.e. Move, Rotate and Scale, can be executed using the same on-screen GUI. These three transformation modes can be accessed using a right-click menu.

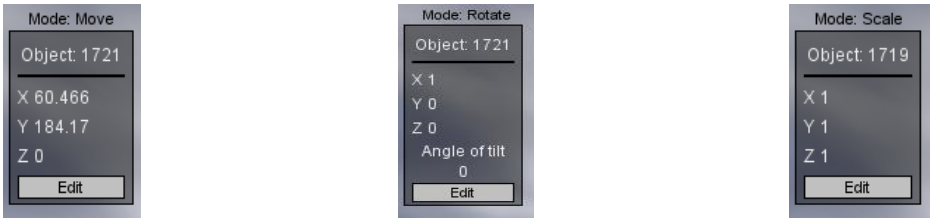

Figure-4 - On-screen GUI to allow for Parametric Inputs for the Move, Scale and Rotate transformation mode

We performed two types of programming: (1) introducing new codes to extend the functionality of TGE to support the CDT-VE, and (2) modifying existing code structures to suit the CDT-VE features.

2.1 Design of the Graphical User Interface (GUI) for the CDT-VE

The Graphical User Interface (GUI) is a combination of the graphics and the scripts that carries the visual manifestation of the CDT-VE. The GUI accepts a designer's control inputs such as mouse movement or keyboard presses. In the CDT-VE, part of the GUI elements includes the designer'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 therefore lines are added or omitted from the scripts themselves. One of the useful development features of the TGE is the visual GUI designer. Hitting the F10 function key invoked the TGE GUI Designer. Using the TGE, GUI design for the CDT-VE was mainly visual through the use of the GUI Designer. Many of the GUI controls were already provided, with some basic properties. Further customizations were performed by adding in 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. Various GUI customizations were done to support the CDT-VE.

The Main Login screen is the first screen designers will see and it is the one of most important features. Since the default state of TGE has no login system in place, new codes have to be developed. The login system allows a designer to host a design session or join an existing one. Care was taken with the design of GUI, the information to be displayed, stored and retrieved from the login screen, the level of security of the design that can be released to a particular designer. A password-protected system was implemented in the CDT-VE for level of authority and access. Figure-5 shows the flow diagram of the login system of the CDT-VE. A designer keys in a username and password and selects which building systems to design. To distinguish between different designers, a designer can pick a color for his Avatar in the VE. Besides the difference in the Avatar's color, the designer's username will also be displayed on top of his avatar's head. On the Main Login screen, a designer can chose to either host a design session or join as a client in an existing one. If a designer decides to host a design session, the designer must select a project (or start a new one), check the "Host Design Session" option and then click on the "Launch Design Session" button. If the design session is password-protected, other designers who wish to join in must use the provided password to log in. If a designer chooses to join an existing design session, the designer must first locate the hosted design session by hitting either the "Query LAN" or "Query Master" button. A search command will be issued to search for any existing design sessions available on the LAN and the internet. Once the respective design session is listed down, the designer simply selects the design session and clicks on the "Join Server" button (Figure-6).

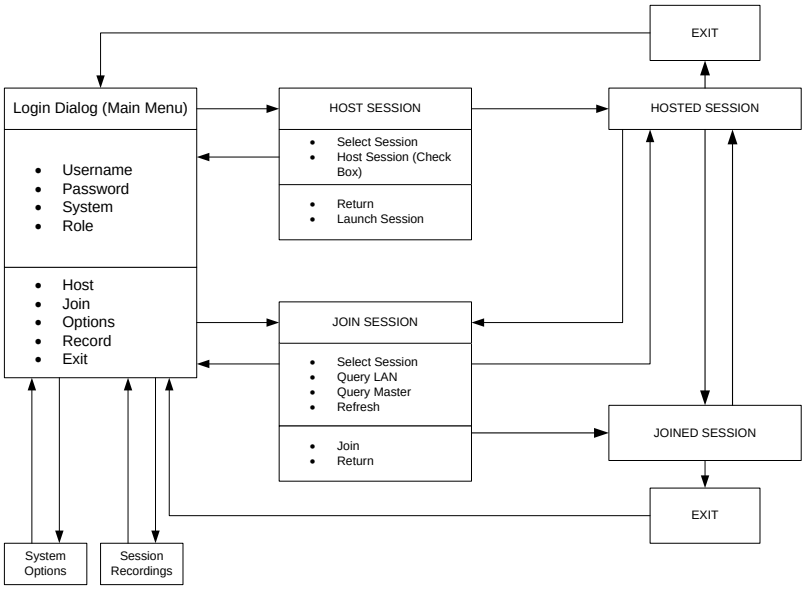

Figure-5 - Flow diagram of the login system

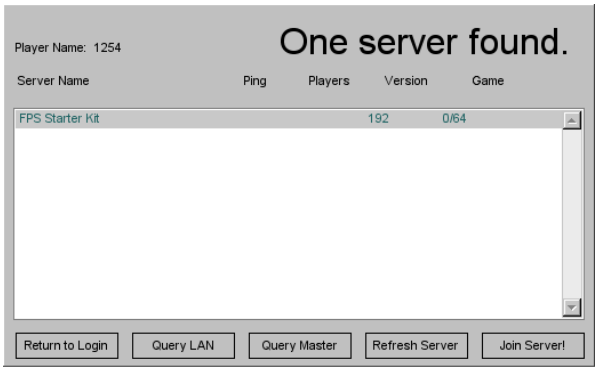

Figure-6 - The join server window

2.2 3D Object Manipulation and the Right Click Menu System

The default implementation of the TGE's World Editor was not intuitive enough as many of the commands were either available only through the main menu, shortcut keys or a combination of both. We believe that some commonly used commands should be easily accessible through the right-click menu. However, we were not sure whether this was possible or not in TGE, as no prior implementation existed in any of GarageGames's products. After several tryouts, the right click menu system for the CDT-VE was developed. We have implemented a 1-level right-click menu system that included the most commonly used commands for 3D object manipulations in the VE. The final implementation of the right-click menu is shown in Figure-7. We plan to further research this GUI aspect to ensure that the most important commands can be easily executed.

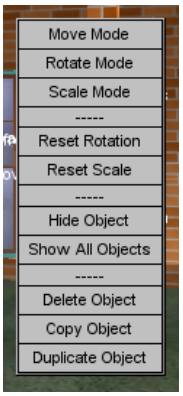

Figure-7 -The right-click menu

The function of each 3D object manipulation features for the CDT-VE is shown in Table.

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 designer.
Reset Scale	Reset the scale of the 3D object into its original state should there be any change in scale made by the designer.
Hide Object	Hide a selected 3D object or selection of objects
Show All Objects	Show all previously hidden 3D objects
Delete Object	Delete a 3D object
Copy Object	Make a copy of the master 3D 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 changed and it is independent of the characteristics of the master object
Duplicate Object	Make an exact duplicate of the master 3D 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 the master door will be reflected onto the duplicated door. This feature allows for mass property change for similar 3D objects.

Table-3 - Description of the commands available through the right-click menu system

2.3 Viewing Modes

In supporting the nature of both local and collaborative design 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 a designer to view objects in the VE from the avatar's eye view. A TPV allows a designer to view objects in the VE from a camera view which is just above and behind the avatar. A BEV took the designer to view from a distance above looking down at the VE. An OV allows designers to circularly view the VE as the center object. BSV was a new viewing mode to accommodate collaborative design among reviewers in a VE. A BSV allows one designer to quickly shift the viewing FOV (field of view) to the location of another designer and have a view just which is behind that reviewer's shoulder. This viewing mode allows a reviewer to view what another reviewer was doing in the VE. The concept of BSV was thought since we found text-chatting and to manually travel to the location of another designer was cumbersome. The Tab key was programmed to accommodate the BSV.

2.4 Collaborative Design in a VE

The CDT-VE allows for real-time 3D object manipulation not only on the local PC, but across the network either through a LAN or the Internet. In our collaborative design session, all forms of data are tracked between all connected PCs. Client PCs can join a Hosting PC (acting as ad-hoc Server) over the internet or a LAN connection

and perform design tasks such as 3D object manipulation in an almost identical fashion to that of the host. 3D object manipulation task includes moving, rotating and scaling objects, creating objects, deleting objects, and copying and pasting objects. These tasks can be executed simultaneously in real-time by all connected PCs, and changes can be seen in real-time on all PCs. We have successfully tested the collaborative design feature with seven client PCs connected to a Hosting Server. In developing the collaborative design features, we made changes and developed new codes to the TGE. The client PC is able to execute the Design mode (World Editor mode) and load a scene just like the host. Once the client PC is in the Design mode, custom codes translates each command that the client performs in the VE back to the server. Each changes 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 (or state) of the object. The server receives these server command messages and executes them in the order they are received. To avoid message overloads (which can affect the real-time element from the VE), the number of messages the client sends out 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. The host can now adjust how many times within a second these updates are sent out to the clients to help preserve the bandwidth if many clients are editing the level simultaneously in real-time. Figure-8 shows a generic representation of the inner-workings of the messaging system of the CDT-VE.

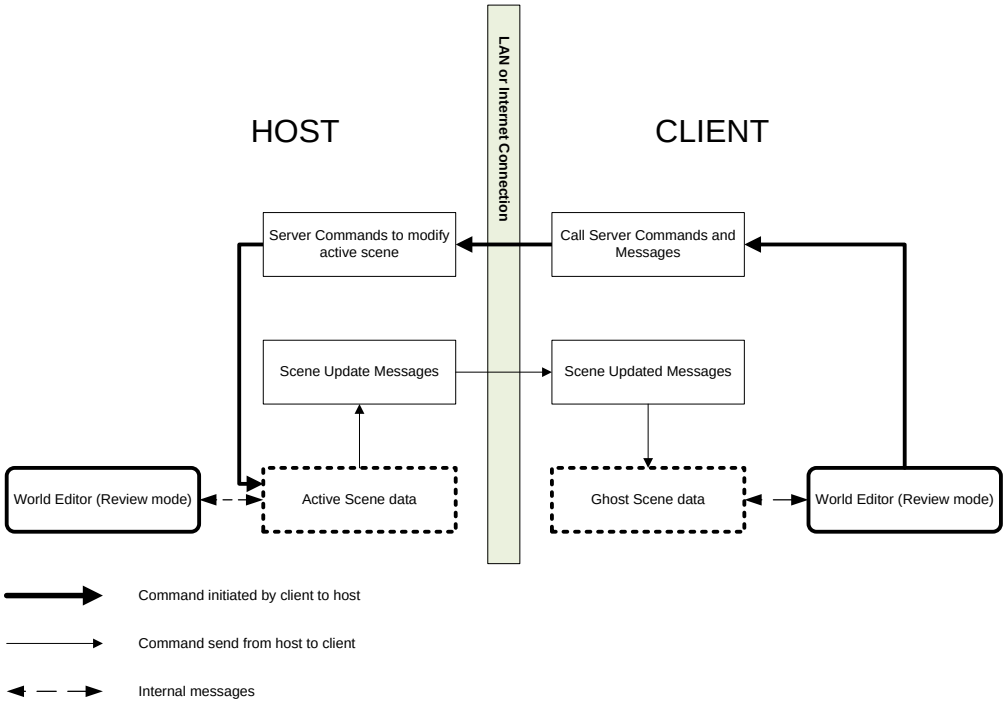

Figure-8 - The inner-workings of the messaging system

2.5 Database Support for the CDT-VE

In supporting design information processing, we decided to use SQLite which is an open-source database software. SQLite is a different from SQL in that it offers no built in server commands. This makes SQLite ideal for applications which already have a networking infrastructure, like the TGE, or applications that require no internet access such as cell phones or mp3 players. SQLite is specifically designed to be added to the source code of a software application to give the application a self-contained, persistent database system which requires no configuration (http://www.sqlite.org). 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 application. TGE lacks the database we need to store design and 3D object information. Compared to other types of integration of database engine, SQLite is the easiest to implement into the TGE. Once SQLite was integrated into the CDT-VE, it created a quick reference, encrypted database system which allowed for many commands such as creating, deleting, querying, editing, and reorganizing of the data. Some of the uses for SQLite that were implemented in the CDT-VE are logging the designer's login information, store, retrieve and display design information and building codes, tracking the object movements in the editor and embed information such as dimensions, material, and cost of the objects in the VE.

Figure-9 shows how information is bi-directionally passed between the designer and the databases. Based on the designer's request for information, it will be processed. The information is passed from the user through the TGE scripting code and then through the SQLite code. 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.

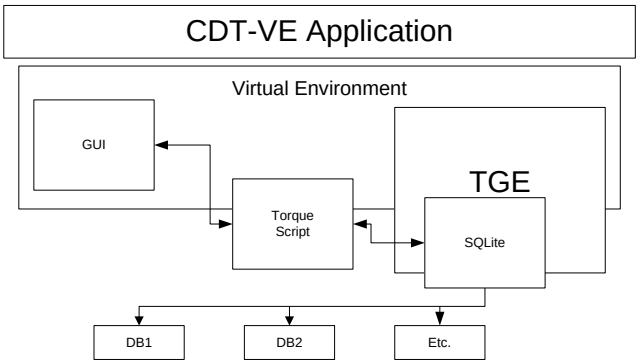

Figure-9 - Data flow diagram for the information passing through one component to another

In the CDT-VE, the designer inputs the information through the GUI system. Editable forms and text boxes, or multi-line editable text boxes were used as the GUI input. This information is then passed to the SQLite engine code in a string format using scripting code. Below is an example of a line of scripting code which might be used to make a query of a database. The '@' symbol represents the concatenation operation (connected or linked in a series) while the '%' before a word represents a local variable.

```
%res = %obj.query("SELECT * FROM data WHERE id= " @ %id @ " AND age<7 AND name LIKE " @ %name);
```

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. 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. Based on the resources available on GarageGames' website, three steps were taken to embed the SQLite database into the CDT-VE: 1) The SQLiteObject.cc, SQLiteObject.h and sqlite.lib files were copied into the solution folder and added to the Visual Studio 2005 IDE compiler solution. Then the CDT-VE has to be recompiled, 2) The SQLite.dll was added to the folder that holds the CDT-VE executable. This step is important for the SQLite engine to properly run within the application. Without this, the application will produce error messages and crashes, and 3) The sqlite.cs file was added from the resource to the list of script files used within the CDT-VE. This allows for the use of simple Torque script commands to execute actions rather than having to know the complicated strings to pass to the SQLite engine. With the three steps described above, SQLite database support was embedded into the CDT-VE.

3 Conclusion

The premise of the CDT-VE is to allow designers to collaborate in real-time in a VE, to share the same virtual design space, irrespective of their physical location. As we mentioned earlier, with respect to the CDT-VE, we will be: 1) doing a study how architecture students collaborate in a real-time VE and 2) defining suitable graphical user interfaces (GUIs) for design tasks in a real-time VE. The study will involve a few experiments and the setup is as follows. Subjects will be broken down into groups of one, two, or three. The subjects who are by themselves will serve as a control group. Subjects who are in groups of two or three will be randomly divided into two sets of testing conditions. We will call these "conditions A" and "conditions B". Subjects under conditions A will be working face to face with each other and use a single non-collaborative computer with the CDT-VE installed. Subjects under conditions B will be working in separate rooms each with a collaborative computer equipped with the CDT-VE, and a third party voice-over-IP program (Figure-10). All groups will be presented with specifications and 2d plans of the 3BR house. We will also demonstrate briefly how to use and navigate in the CDT-VE. Subjects will then be asked to construct the 3BR house in the VE using the CDT-VE. Note that only groups working under conditions B will actually use the collaborative capabilities. All virtual resources necessary to complete the task will be provided. The

CDT-VE will automatically record the actions of each subject, as well as log technical data (such as number of moves, rotations, etc). The conversations of subjects under conditions B will be recorded, and videos of those under conditions A will be recorded. The data will then be analyzed to determine the effects of working collaboratively in a VE as opposed to working face to face. We hope to publish our findings once we have our data analyzed.

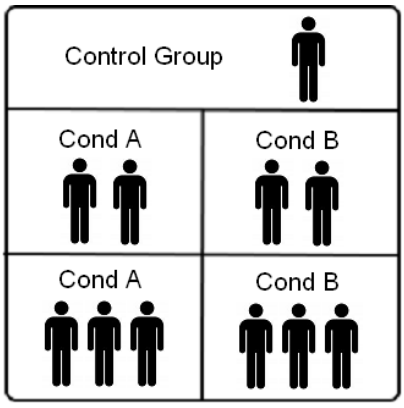

Figure-10 - Proposed testing condition

4 References

Bryson, S., (1996) Virtual Reality in Scientific Visualization. *Communications of the ACM, 39(5),* 62-71.

Howard H.C., Levitt R.E., Paulson B.C., Pohl J.G. & Tatum, C.B. (1989). Computer integration: reducing fragmentation in AEC industry. *Journal of Computing in Civil Engineering, 3(1),* 18–21.

Liston, K., Fischer, M., & Kunz, J. (2000). Designing and evaluating visualization techniques for construction planning. In Renate Fruchter, Feniosky Pena-Mora, & W.M. Kim Roddis (eds). *Proceedings of Computing in Civil and Building Engineering (ICCCBE-VIII), 2,* 1293-1300.

Maher, M. L, Cicognani, A., & Simoff, S. (1996). An Experimental Study of Computer Mediated Collaborative Design. *Proceedings of the 5th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'96),* p 268.

Schuckmann, C., Schummer, J., & Seitz, P. (1999). *Modeling Collaboration Using Shared Objects*. *Proceedings of the International ACM SIGGROUP on Supporting group work*. ACM Press.

Shiratuddin, M.F. & Fletcher, D. (2006), Southern Miss' Innovation and Commercialization Park: Development of a Large Scale Real-Time Virtual Reality Environment, Proceedings of 6th International Conference on Construction Applications of Virtual Reality, August 3-4, 2006, Orlando, Florida, USA.

Shiratuddin, M.F., & Thabet, W. (2003). A Framework for a Collaborative Design Review System Utilizing the Unreal Tournament (UT) Game Development Tool. Amor R (ed). Proceedings of the CIB W78 20th International Conference on Construction IT, Construction IT Bridging the Distance. CIB Report 284. 310-318. Retrieved March 21, 2008 from the ITC Digital Library Website: http://itc.scix.net/cgi-bin/works/Show?w78-2003-310

Theoktisto, V., & Fairen, M. (2005). Enhancing collaboration in virtual reality applications. *Computers and Graphics, 29,* 704–718.

Whyte, J., Bouchlaghem, D., Thorpe, A., & Mccaffer, R. (1999). A survey of CAD and virtual reality within the house building industry. *Journal of Engineering Construction and Architectural Management, 6(4),* 371-379.