Summer 2003

3D User Interface for a File Management System

Luiz Fernando Capretz
University of Western Ontario, lcapretz@uwo.ca

David Carter
CGI, david.carter@cgi.com

3D User Interface for a File Management System

by David Carter and Luiz Fernando Capretz
University of Western Ontario, London, ON

Abstract

Two-dimensional graphical user interface (GUI) is now firmly established as the preferred interface for most applications. The purpose of this work was to develop a three-dimensional user interface as a front end for a file management system and to evaluate the efficiency of a practical 3D application. In order to create this software, a previously defined 3D graphics engine, called Valve Software's Half-Life, was extended to provide a directory traversal and the basic file management functions (cut, copy, paste, delete). The project was divided into two basic components: generating the 3D "world", and altering the Half-Life engine to provide some features of file management.

Sommaire

L'interface utilisateur graphique bidimensionnelle est l'interface de choix pour la plupart des applications. Le but de ce travail était de développer une interface utilisateur tridimensionnelle (3D) pour un système de gestion de fichiers et d'évaluer l'efficacité d'une application 3D. Un moteur pour graphiques 3D du nom de 'Valve Software's Half-Life' a été amélioré. Ce dernier comporte maintenant une traversée de répertoire et des fonctions de base tel que couper, copier, coller et effacer. Ce projet a été divisé en deux parties : génération du "monde"' en trois dimensions et modification du 'Half-Life engine' dans le but d'offrir quelques caractéristiques d'un gestionnaire de fichiers.

1.0 Introduction

The graphical user interface (GUI) was developed in the late seventies, successfully commercialised in the early eighties, and has since become the integral part of every modern operating system.

Today's computer users are presented with an ever-expanding amount of information. Due to this increasing volume of data, the user eventually becomes overwhelmed with it. Three-dimensional user interfaces cater for the organization of this data and can help the users regain control over the information in a natural way [1].

Currently computer users are bound to the "desktop" metaphor. This metaphor's longevity is a testament to the strength of its design. Of course, over twenty-some years improvements have been made to that original design, but the basic elements (e.g. icons, pop-up and pull-down menus) that implement the *what-you-see-is-what-you-get* idea have remained the same.

Just as decreasing hardware price and increasing hardware capabilities made the pseudo-2.5D GUI affordable in the eighties and widespread in the nineties, these hardware trends will make 3D GUIs affordable in the near future. GUI with 3D capability offers great potential for improvement over today's 2.5D GUIs (in which the xy-plane is clearly defined and the appearance of depth is created through the obscuring of "background" windows). For instance, multiple overlapping windows are hard to identify, iconified windows more so, and one quickly runs out of space trying to group related applications; alleviating the problem of window trashing is one of the main design goals of a 3D-GUI.

The 3D interface is not a new thing [3, 4]. They have existed ever since the first 3D object was output by a computer. Leach et al. [2] presents a metaphor used for a 3D-GUI in which windows are arranged in a *tunnel*. The user is positioned in the middle of the mouth of the *tunnel* looking toward the other end. Windows are displayed with a perspective projection. In addition to the front-end window, there is a "hanging" mode where the windows are hung on the left or right wall of the tunnel. Another new idea is a 3D cursor with six degrees of freedom, called the *magic wand*, which "floats" over the top of objects rather than being part of the screen.

The purpose of this research was to develop and evaluate a 3D user interface as a front end for a file management system (such as Microsoft Windows© Explorer). The implemented software reused a modified version of the user interface used in Valve Software's Half-Life engine.

To make this sort of interface popular, we face two problems: First, within the operating systems domain, the use of 3D graphics is heavily under-used. Secondly, for ordinary users, on-the-fly creation of 3D "worlds" is a totally new trend. However it will soon be available to personal computer users.

2.0 File Manager Requirements

Although it seems reasonable that the home users should be able to fully immerse themselves in a 3D rendered world, it is not always possible to efficiently provide this. Typically, 3D interfaces are often awkward for the developers to represent data, and harder for the users to manipulate than the standard 2D user interface.

In considering 3D graphical user interfaces, it became evident that if an effective 3D user interface were to be developed it needed to be the front-end for a practical application. Thus the idea was to provide the user with a system for managing files, one that incorporated 3D rendered graphics, yet at the same time maintained the same (if not a higher) level of usability as the standard 2D interface to which the home user had become accustomed.

The developed software was intended to be an extension of the standard Microsoft Windows© Explorer. It enables the user to traverse a file system from a first person perspective using a 3D interface, and it provides the functionalities listed in Table 1.

Table 1: Functionalities of 3D Interface

File Manager	User Interface Functionality
Create New Directory	Generate the User Interface
Rename	Inventory
Cut	Return to Root
Copy	Map Layover
Paste	Select All
Delete	Refresh

This set of file management functionalities was provided as they were considered to be the most common tasks performed by the average user. The user interface functionalities were given to aid the user in performing the provided file management functions.

2.1 The User Profile

The software is intended for two different types of users. The first type is the person unfamiliar with the file structure employed by an operating system and who also has trouble visualizing the file system in a two dimensional manner. For him/her, the software is useful since it creates a structured environment that is analogous to a file system and also it allows for a visual learning process to commence. The second type of user is the one familiar with first person perspective employed in many

games; for him/her, the software becomes an interactive means of monitoring the user's file system.

2.2 The User Interface

The software implements a modified version of the Half-Life engine and is employed to represent the directories and files contained within the file system itself.

A directory is represented by a rectangular room. Along two of the four walls in the room transporters to the subdirectories are placed, and along one of the remaining walls a transporter to the parent director is also placed.

Files were originally intended to be represented as their native Microsoft Windows© icons; however, as development progressed, the need for a more reusable representation arose, and files are represented as panes of glass hovering in their appropriate room.

3.0 The Design of 3D File Manager

Originally it was thought that the software would consist of three main components: the file manager, the 3D user interface, and the Half-Life engine. However, as the design was refined portions of these components shifted, and a more efficient design emerged.

Since the Half-Life engine follows the client-server architecture, it was able to absorb the other two components. The file management component was consolidated with the server side, while the client side absorbed the user interface functions. However, it was also necessary to develop software to generate the data needed for the Half-Life engine to represent the file system. Thus the system continued to contain three components.

As depicted in Figure 1, the *InitLevel* class is responsible for generating the data needed for the Half-Life engine. The *Half-Life engine* package contains both the server and client components. The *DirectoryInfo* and *FileInfo* classes are taken from the .NET framework and are used to aid in the file system traversal.

In using the Half-Life engine, it was possible, by reusing the necessary portions of the engine and then adding the file management and user interface functions, to produce the desired 3D user interface. However, in using the Half-Life engine, the major stumbling block became the compatibility issues between Microsoft Windows© and the engine (an OpenGL based system).

4.0 Implementation

The implementation of the 3D file management system took place in two phases. First, the map generator (implemented in Visual C++ .NET) was developed to build the virtual world the user could walk through. Secondly, to add the file management capabilities promised in the requirements, alterations were made to the Half-Life engine.

In originally creating the design for the 3D file management system, it was not clearly understood what was and was not possible with the available development software. That is, a map was generated by the system after recursively searching the directories within it. That map was compiled and passed off to the Half-Life engine, where it was rendered so that the user could traverse it and perform the specified operations.

However, the details of each of these functions altered greatly as three concepts became clear:

1. The ease with which the .NET Framework was able to traverse the file system to provide the basis for the maps.
2. The complexity of generating the map files. Even though before they are compiled the files are plain text, the difficulty in creating these maps, in terms of their size and the order in which each of the brushes or entities must be added became increasingly apparent.
3. The sheer size of the Half-Life engine became overwhelming. The complexity of the source code, in terms of how the various modules of the engine communicate, made understanding it and adapting it rather difficult.

In order to adapt to the changing climate of the project, it was necessary to alter - and in some cases overhaul - the design that was originally created. However, in doing so, the opportunity for constant inspections and revisions were created.

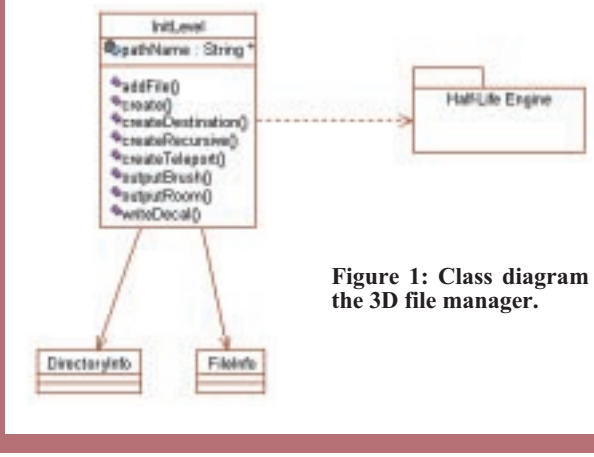

Figure 1: Class diagram for the 3D file manager.

4.1 Communication with the Half-Life Engine

The Half-Life engine is separated into two main components: the "client" and the "server". The "client" basically refers to anything that occurs on-screen (player movement, targeting, animations, etc.…) and the "server" side performs the background functions (entity AI, trajectory calculations, etc.…).

The server is able to send messages to the client by using the following set of macros:

BEGIN_MESSAGE(), WRITE_BYTE(), WRITE_STRING(), WRITE_SHORT(), WRITE_LONG(), WRITE_CHAR()

The server can also receive messages from the client by using the macro:

CMD_ARGV()

The client sends message to the server by using the function:

ClientCmd()

The client reads from the server by using the following macros:

HOOK_MESSAGE(), BEGIN_READ(), READ_BYTE(), READ_STRING(), READ_SHORT(), READ_LONG(), READ_CHAR()

Although each of these functions exist, in server to client communication the data types used for the "write" and "read" operations must be synchronized or the communication does not work.

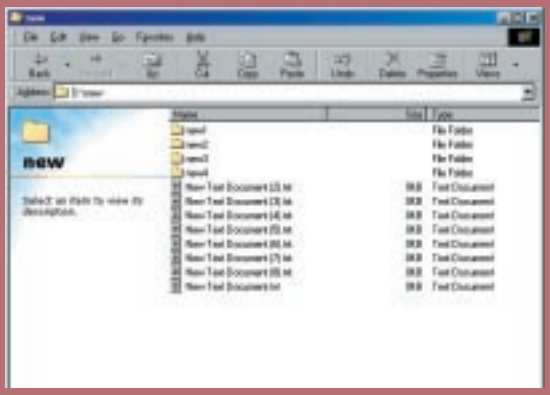

Figure 2: Directory listing of "new" directory.

Figure 3: "cut" is selected from the right-click menu.

5.0 Example of Use

The following sequence of steps describes an example of the "cut" operation. Figure 2 shows a directory listing of the "new" directory, containing the subdirectories "new1", "new2", "new3", "new4", and eight "new text documents". Figure 3 displays one of those text documents being selected, using the 3D user interface. Note the two portals in the background for subdirectories "new3" and "new4". Figure 4 depicts the selected text file being removed from the directory. Finally, Figure 5 shows an updated listing of the "new" directory after the "cut" operation, confirming that there are only seven text documents in the updated directory after "New Text Document (2) has been deleted.

6.0 Conclusions

This paper shows the results of an attempt to create an application that contains an efficient 3D application. The process of creating the 3D file management system started with the identification of a gap within the computing industry; that is, a lack of 3D user interfaces incorporated into an operating system, even though the hardware to enable this exists. Computer games provide compelling evidence that desktop computers are capable of supporting interactive three-dimensional visualization, yet 3D interfaces remain largely tied to niche markets such as CAD/CAM

Having identified this problem, it was necessary to come up with some important requirements for a system that would utilize a 3D GUI but would be based on a piece of software that was currently in use. After much thought it was decided that a file management system would be the best system to implement, as it contains features most users are

Figure 4: The selected file is removed.

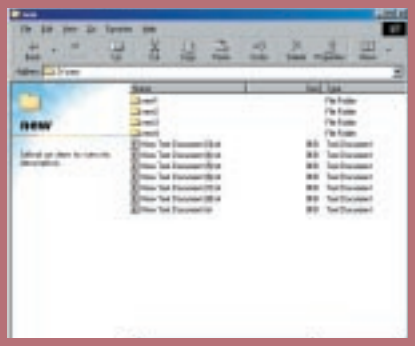

Figure 5: The file is removed from the directory.

aware of and it allows for a friendly 3D GUI to be developed for it.

The design and implementation of the system originally started out as two separate activities plus the expansion of a 3D graphics engine. As the project progressed, these three tasks became intermingled and a very iterative process was followed.

Testing revealed the limitations of the software as it relates to the 3D engine used. Unfortunately the 3D engine chosen placed limitations on the maximum number of directories that could be rendered in the 3D user interface. However, this limitation leaves open the opportunity to develop a better engine to use as the backbone of the system.

7.0 References

[1]. Robbins, D., "Thoughts on the state of 3D in our industry",<http://research.microsoft.com/~dcr/talks/3dui_and_industry_02.htm>, May 2003.

[2]. Leach, G. et al., "Elements of a 3D Graphical User Interface",<http://goanna.cs.rmit.edu.au/~gl/research/HCC/interact97.html>, March 1997.

[3]. Poupyrev, I., "Research in 3D User Interfaces", *IEEE Computing Society Student's Newsletter*, 3(2):3-5, 1995.

[4]. Cockburn, A. and McKenzie, B., "3D or not 3D? Evaluating the Effect of the Third Dimension in a Document Management System", in *International Conference on CHI-2001*, ACM Press, New York, pp. 434-441, 2001.

About the authors

David Carter received a bachelor degree in Software Engineering from the University of Western Ontario in 2003, and is currently pursuing his B.Sc. in Applied Mathematics. He was awarded the Western Scholarship of Distinction in 1999 and has been in the Dean's Honour List for the last two years. His 4th year capstone project was among the finalists in the Engineering Design Contest. His research interests include: human-computer interaction, three-dimensional graphical interfaces and computer algebra. Mr. Carter is an IEEE student member, and can be reached at dcarter2@uwo.ca.

Luiz Fernando Capretz received his Ph.D. from the University of Newcastle upon Tyne (UK), M.Sc. from INPE (Brazil), and B.Sc. from UNICAMP (Brazil); all degrees in computer science. He has worked at both technical and managerial levels, taught and done research on the engineering of software in Brazil, Argentina, England and Japan since 1981; and co-authored a book in the area entitled: Object-Oriented Software: Design and Maintenance. In the Faculty of Engineering at the University of Western Ontario, Dr. Capretz teaches software design in a new program that offers degree in software engineering. His main research areas include: software product lines, software process, and human factors in software engineering. He is a senior member of IEEE, and can be reached at lcapretz@eng.uwo.ca.