Abstract

The problem of creating an easy way for a user to create scenes containing three-dimensional objects was explored. The project enables three-dimensional scene creation. A three-dimensional object editor was developed. Graphical objects are interactively created, transformed, and plotted in perspective in three-dimensions. Hidden surface removal is implemented using a modified painters algorithm. The emphasis is on an object-oriented approach where the user is unaware of the underlying low-level structure such as vertices, edges, and faces, only having to draw. Objects created using the editor can be saved to disk, read back in, and combined into scenes. Throughout the development efforts were made to utilize the specialized graphics hardware of the Amiga computer.
CHAPTER 1 - INTRODUCTION
Introduction

The project's culmination is a program that allows a user to create objects by drawing, and to combine one or more objects into a final scene to be rendered as solid objects viewed in perspective with nearer objects obscuring those farther away. Originally, the development of a graphical description language was considered. Such a language would require the user to write programs with commands such as:

\[ \text{LINE}(0.8738, 0.00383, 0.9187, 0.987, 0.0003, 0.988) \].

The decision was made to pursue the interactive drawing approach because it would be easier for the user.

The Windows

The project consists of two parts, the editor, and the renderer (Figure 1.1). Each part shares a common user interface in which the user uses a mouse to communicate his desires. The mouse can be used by the user to communicate positional information to the software by moving the mouse to position the mouse cursor, a graphical object such as a picture of a hand, or of crosshairs, which moves on the screen in tandem with physical movement of the mouse. In conjunction with the positional information, the user can press one of two buttons on the mouse. The left button is used to select. By pressing the left button the user tells the software that the current mouse cursor
Figure 1.1 - Windows
position has some context-sensitive importance. The right button is used for menu selection. Pressing the right button enables the user to cause menus to drop down from the top of the screen, and by positioning the mouse cursor on one of the menu selections the user can choose it.

Both the editor and the renderer open a window, a virtual screen which the software can draw into without affecting the contents of other windows. Each window has its own menus. In addition, each of the windows has gadgets, a system-provided graphical object able to accept user input. The gadgets used in this project are called Boolean gadgets which function as buttons. The gadgets are represented as small (approximately 3/8") squares which the user can "press" by selecting with the left mouse button. The software receives a message about the selection, and then can take some context-sensitive action.

Both the editor and the renderer deal with segments, also called, interchangeably, graphical objects. The editor is used to create and modify them, and the renderer to collect them together into scenes and display perspective views of them as solid objects.

The editor presents a window with a title/menu bar at the top from which menu selection can be made, four equal size views, (like small windows in the large window; see figure 1.1), consisting of three drawing areas: an area continually displaying a perspective view of the object as it is created, and gadgets down the left side of the window used to rotate, scale and
translate the object. The three views used for drawing represent the object as seen from the front, the top, and the side.

As drawing is done in one of the views, the result is reflected in all four views. The user draws by positioning the mouse cursor, pressing and holding the select (left) button of the mouse and dragging (moving while holding the select button) a line to the desired second endpoint. As the user draws, a line is drawn continually between the starting point and the current mouse cursor, allowing the user to see what he is doing. The user can then drag a line from the end of the first to a new location, and by continuing in this manner, draw an object. The ability to draw in the three (top, front, and side) views gives the user the ability to create three-dimensional objects.

Once an object is created the user can make menu selections to save the object to disk, to rename the object, or to load a previously drawn object from disk for further editing.

An object can be transformed, i.e., translated, scaled along the three \((x, y, z)\) axes, and rotated about any of the three axes, by pressing gadgets. The result of this is reflected in the perspective view. For each axis there two gadgets allowing the user to scale up or down, two gadgets to translate "right" or "left", and two gadgets to rotate clockwise or counterclockwise.

The rendering window is used with existing objects to combine objects into a scene, and display a perspective view of the scene. The user uses a menu to tell the software he wishes
to load an object, and the software loads it for him. This window also has transformation gadgets which can be used to position a wire-frame representation of the object (a perspective rendering of the outlines of an object using line-drawings). The user can then load another, and by continuing in this way, build a scene. Once a scene is built, the user selects Render from the window's menu, and the software builds a solid perspective rendering of all the objects with hidden surfaces appropriately hidden by closer surfaces. Further information on the user interface can be found in Chapter 2.

The Problems

As with any program development project problems arose. The two things that became the largest problems, and thus the subject of the most intense research in the project were both due to the decision to take an interactive approach to scene development. It seemed desirable to have the computer take as much of the burden off the user as possible, and this necessitated having the computer automatically determine when a face of an object had been completed. Creating a routine to do this took up most of the development time of the project. The other problem was due to the impreciseness of floating point representation. The problem of the inaccuracy of floating point geometric computations is a subject of ongoing research (MATH89). In this project, pixels on the screen rarely corresponded with any
exactly representable floating point number and vice-versa. This made it quite difficult to determine when the user intended to add an edge to an existent vertex, and also complicated the face detection algorithm. These problems are further detailed in Chapter 3.

The project was programmed in the C programming language, and in 68000 assembly language. Since the routines are intended to form a general purpose library of routines, 68000 assembly language was used predominantly for the lower level routines, and C used for the higher level routines. All assembly language routines were provided with C bindings (small assembly language routines to act as an interface between C language calls and the actual assembly language routines).