Design of a Telescope Control System Interface

T. Hudson
Computer Science Department
Sam Houston State University
stdtgh13@shsu.edu

ABSTRACT

A telescope control system interface provides a means of communication between an astronomer, or telescope operator, and a telescope. This communication loop has historically been unfriendly to the average user, often relying upon text-based screens or very basic graphical user interfaces. In this paper, we present a new standard for the telescope control system interface by utilizing an advanced graphical technology. Specifically, we intend to code an application in C++ that employs the OpenGL API. This allows us to present information in a manner that is concise, flexible, and easy to learn.

1. INTRODUCTION

I and my mentor, Dr. James Fowler, decided to attempt this project together, with the hopes that we might harness the power of computer graphics and apply it to a field of software in dire need. The explosion of computer graphics technology within the last fifteen years has revolutionized many industries. Computer aided drafting has become an integral part the engineering industry and computer animation has become the new love of the movie industry. The military makes use of realistic computer simulations while hospitals use three-dimensional imaging to analyze a patient. With this paper, we intend to show how computer graphics technology will revolutionize the modern telescope control system interface.

With graphics technology, we can create a visually pleasing system that accomplishes several goals: An interface should be extremely intuitive, so as to minimize the amount of training time needed. A well-designed GUI should require little to no instruction prior to operation. It should also make an efficient use of visual space on the monitor, and cut down the number of total monitors needed for telescope operation. It should be easily customizable for the various types of users. It should be as self-monitoring as possible, requiring the user's attention only when absolutely necessary. It should minimize the amount of user error by requiring as little input as possible.

To make all of these goals attainable, we decided to use a graphical standard known as OpenGL. This API allows us to achieve our goals with graphical operations that would otherwise be very difficult to implement. An added benefit to using OpenGL is hardware support. Most of today's computers have dedicated GPUs that support at least a portion of the OpenGL API in hardware. This allows for added graphical effects without a significant burden being placed on the primary CPU.

2. BACKGROUND

In this section, we intend to give some background about our current telescope control system, and the tools we intend to use.

2.1 Current Telescope Control System

The telescope control system we used as a model for a system that needed improvement was at the Hobby-Eberly Telescope (HET) at McDonald Observatory in the Davis Mountains of West Texas. The need for improvement derives from the fact that the current TCS interface was borrowed from that of another telescope. It was not designed with the HET in mind. The main window contains mostly text, the majority of which is not needed. Dr. Jim Fowler suggested a graphical interface that did not include unneeded information.

To accomplish this, we decided that OpenGL was the best choice. Introduced in 1992 by Silicon Graphics Incorporated, it is the API of choice for programmers who want to code graphically intense applications. It is a mature API that is supported by multiple platforms with multiple languages. We also decided to use C++ because it is extremely portable and a language we are both comfortable with.

3. DESIGN

In this section, we will discuss the design issues of our graphical telescope control system interface.

3.1 Graphical Features

In order to achieve our design goals, we utilized a variety of graphical features readily available in OpenGL.

3.1.1 Three-Dimensional Representation

OpenGL has both 2D and 3D applications. The latter, however, is arguably the prime focus of the API. By allowing the programmer to easily construct geometric primitives in memory, OpenGL facilitates the assembly of entire 3D environments. OpenGL then handles the math associated with transforming the 3D image onto a 2D screen. This would allow any telescope control system to contain a 3D model of the dome and telescope structure. Or perhaps the astronomer would prefer some sort of 3D finding chart. Even the core of the interface could be represented in 3D if desired. In any case, 3D representation often allows for quick visual information retrieval. For example: Instead of finding the dome shutter status in a list of numbers, it is much easier to look at a picture. Not only would you instantly know the status of the shutter, but also the dome's orientation.

3.1.2 Rotatable and Scalable Text

While many GUIs may boast text than can be rotated and scaled, they do not do so with the precision and ease of OpenGL. By representing text with geometry, OpenGL can scale and rotate text at infinitely precise increments. This can in turn, make a telescope control system interface infinitely customizable.

3.1.3 Pop-ups and Mouse-overs

This feature is not specific to OpenGL, but plays an important role in the efficiency of the GUI. By hiding text information until it is needed, the overall size of the GUI can be reduced. Mouse-overs are important because you can obtain additional information about any object on the screen by simply placing the cursor over the object. Pop-ups can be utilized to warn the telescope operator of errors or other immediately pertinent information. This negates the need for a dedicated error message space.

3.1.4 Alpha Channel Blending

For every visual element of an OpenGL program, color information is stored in four discrete channels. These channels are referred to by the acronym RGBA (Red, Green, Blue, and Alpha). The first three channels determine the overall color of the element, while the alpha channel determines the opacity of the element. An object drawn on the screen at only half opacity would reveal any objects drawn below it. This can effectively allow the simultaneous display of two visual elements that happen to occupy the same space.

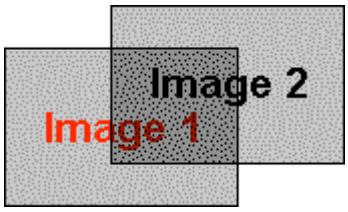

Figure 1 - A semi-opaque window achieved through alpha blending

This could be applied to messages or windows so that they are drawn over information on the screen without completely obscuring it.

3.2 Modularity

A major goal of ours was to make a telescope control system interface that could easily be changed to suit a particular observatory and even particular users. By containing each piece of information in a module, and allowing the configuration of each individual module, we can produce an extremely flexible GUI. OpenGL allows for "subwindows" within windows that can be resized and reshaped. Within these subwindows, we can place any kind of information. For example: In one module, we might put temperature information. The text size and colors within that module could be arranged in any fashion. Then the module itself could be dragged to any desired position on the screen. It could be resized to a very small box, so that it did not take much space, or it could be expanded to fill the entire screen. In this manner, any number of modules could be added or removed from the TCS, making it easy to upgrade in the future. For our telescope control system interface, we added a temperature module, a text module, a history log module, a dome status module, a radar map module, and a tracker module. Given that our telescope control system interface is very simple, it does not include modules for calibration, scheduling, guidance, data information, or maintenance. All of these functions are already managed by other programs at the HET, however they would not bee too difficult to add. After arranging modules in a suitable manner, the arrangement can be saved, so as to personalize the TCS for any number of users.

3.3 Implementation

In this section, we discuss the way our telescope control system interface was implemented.

3.3.1 Planning

Dr. James Fowler made the bulk of the decisions concerning the design of the interface. He suggested a circular design to correspond with the degree of motion of the telescope. We also incorporated many elements from the current TCS interface, but attempted to streamline them in the process.

3.3.2 Interviewing

To ensure satisfaction by our end-users, the telescope operators, we conducted many interviews. We queried each operator about the feasibility of various design elements. We also obtained many new design elements from the operators themselves. We were also made aware of the various presentation needs of each operator.

3.3.3 Programming

In order to take advantage of OpenGL's capabilities without worrying about the intricacies of window-system dependent commands, we employed the help of GLUT, an OpenGL Utility Toolkit written by Mark Kilgard. It allows us to start an OpenGL capable window without regard to a specific operating system. We constructed the bulk of the program with C++ and OpenGL, although we also made use of system calls to a PERL script made to obtain POSS II images from the web. Over the 10-week period, an interface gradually evolved that included most of the aforementioned design elements. Though not fully operational, it was a sufficient mockup to determine the effectiveness of such an interface.

4. CONCLUSION AND FUTURE WORK

Within the given time constraints, we were unable to implement all intended features of the interface or implement connectivity with the TCS server. In the event that we were given more time to complete our project, we would like to implement more advanced features of OpenGL, such as texture-mapping and more 3D features.

We did a few follow-up interviews with telescope operators to get their reaction to the new interface. We were pleasantly surprised by the response we received. The graphical features and practicality of our new interface were very well liked.

We believe that we have uncovered the mere tip of the iceberg of what is possible, and we are excited to see what the future has in store for the modern telescope control system interface.