Exploration of a 3-D World

Zachary R. Greer
TJHSST Computer Systems Lab
Alexandria, Virginia

June 5, 2009

Abstract

The project opens a display which allows the user to explore a minimal 3-D world using OpenGL. A grid of spheres are provided as reference. Mouse input is taken to modify the direction in which the user is looking, and keyboard input is taken for movement across a plane. Movement occurs with respect to the direction the user is looking along the xy-plane. Spherical coordinates are used to achieve this intuitively and easily.

1 Introduction

The main objective of this project in its current state is to create a 3-D environment that can be explored by the user. C was chosen to implement this because of its speed (Norvig, Fig. 1) and the OpenGL libraries it offers. Though other languages do offer graphical libraries, none are nearly as fast as C. C++ was discarded because of its constantly changing syntax and the author's unfamiliarity with said syntax. Minimal though the features in this environment are, the goal is achieved. The user can move freely about on a plane, and can look around as she wishes. Intuitive mouse control and key layout are used for input.

Perspective and movement are implemented with spherical coordinates. This system translates easily to the controls, makes movement simple, and also describes the line of sight vector in a simple way for OpenGL to understand.

There are plans for networking, surface loading, physics, and a proximity-based chat system as well.

2 Background

The chat system I plan to use was inspired by "Proximity-based chat in a first person shooter: using a novel voice communication system for online play", by Gibbs, Wadley, and Benda, concerning the chat systems used in games. Namely, they generally consist of a two way radio systems, which are not very conducive to the social aspect of the game. The article suggested instead employing a proximity-based system of chat, much like we see in real life. The study they did suggested that this enhances social, teamwork, and realism aspects of gaming, all extremely important in their own right. The plan is to use this system to attain precisely that effect.

"How computer gamers experience the game situation: a behavioral study", by Clarke and Duimering, has varied suggestions on the structure of game, including controls, networking issues, and players' reactions to gaming environments, as well as discussing players' experience of the game, styles of play, and frequency of play. The study was done by interviewing FPS players of different skill levels about their in-game experience. These will be important considerations while constructing my project, as the users' experiences should be as realistic and enjoyable as possible.

OpenGL is employed as the graphical interface. OpenGL is a free, open-source graphics interface compatible with C, and possesses powerful 3-D capabilities not seen in other graphical frontends.

Spherical coordinates are used to track and display the user's line of sight. As will be seen in the Development section, this is the best way to store and modify this data, for input, line of sight, and movement reasons.

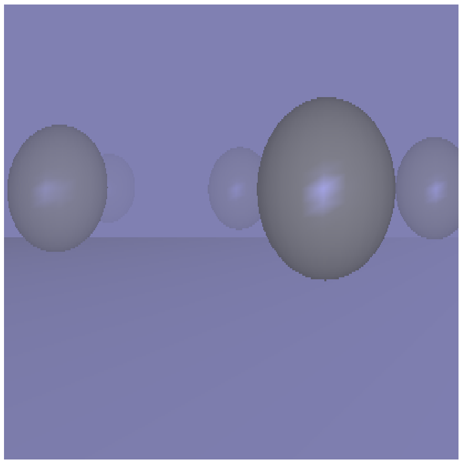


Figure 1: This is how the program currently looks.

3 Development

3.1 Current State of Development

In this stage, the program allows the user to explore a 3-D environment with a grid of reference objects and a floor, which move dynamically to where the user needs them, and takes key and mouse input. It displays fullscreen with realistic lighting and fog, and recenters the mouse to allow the user to look anywhere without worrying about screen constraints.

3.2 Perspective

The first real issue presented to one trying to simulate a person exploring an environment is that of perspective. When the mouse moves to look around, what should be displayed? Thankfully, OpenGL handles some of that, in that it will look from one point towards another point. Still, how do we best represent this? Rectangular coordinates, the ever-default of graphing systems, lend themselves to this conundrum rather poorly, due to the extreme incapacity in the realm of circles, which is what turning really is. The king of 3-D circular systems is, of course, the spherical coordinate system. I use spherical coordinates to determine the direction in which the player is looking. OpenGL asks for a vector to the point you want to look at in Cartesian coordinates. To achieve this, I store a phi and a theta for the user, as this is the most intuitive way to describe how you look around.

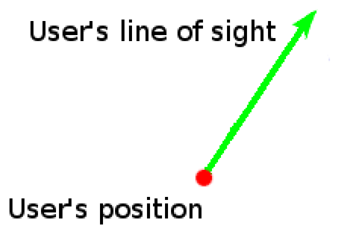
We know the vector in which the player is looking, and the player's position. How do we describe this as a point to which OpenGL can look at?

With the radius equal to 1, the spherical format can easily describe a point to look at. In $< x, y, z >$ format, the point is:

$$< x_v, y_v, z_v > = < x_p + cos(\theta) * cos(\phi), y_p + sin(\theta) * \tag{1}$$

where $< x_v, y_v, z_v >$ is the position vector of the point you want to view, and $< x_p, y_p, z_p >$ is the position vector of the player. One of the reasons this is so useful is that it translates easily to the mouse. If the mouse moves right, increment θ. If it moves left, decrement θ. If it moves up, decrement ϕ. Down, increment ϕ. The other use of this system is that for moving, to obtain an $< x, y >$ unit vector in the direction you want to travel, you just take $< cos(\theta), sin(\theta) >$.

Another upside to this system is that the



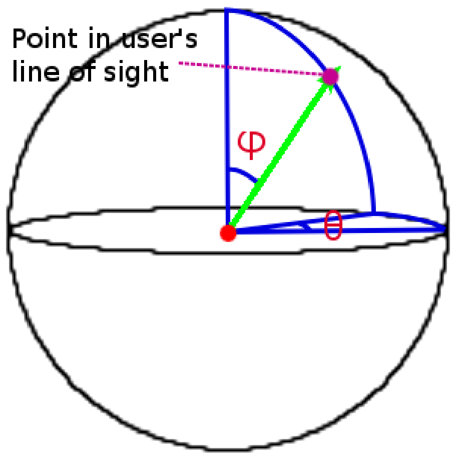


Figure 2: Shown is the user's position in a sphere, with the relevant angles labeled.

entire line of sight vector (and motion vector) can be described with two variables, which are quickly modified and also quick to produce either vector.

3.3 Working with X11

I was presented with a surprising amount of difficulty over the course of the year trying to use particular parts of the user interface. The first issue was fullscreen display, as OpenGL needs a connection to the X server, and that's not as easy as it sounds. If you accidentally open too many connections, your program dies. Once I finally realized what was happening, I found a way to ensure only one connection was opened. The next issue comes with the mouse controlling look direction. It was simple to remove the mouse from view, but how do you stop it from hitting the edge of the screen? It turns out that there is a command in X11 to move the cursor (called XWarpPointer), but then it generates a key movement in OpenGL, indistinguishable from the user's movement

3.4 Extendability

The program is equipped with variable names that are intuitive, and is commented and organized in such a way that the program is easily changed to fit one's needs. This software can be included in a larger program that causes models one loads to move, and the program will display the new situation with no added code. This program essentially handles all of the graphics problems one would run into. It is also extremely modular, (at least as best as C can do) so could be easily integrated into other applications.

4 Analysis

At the moment, I have very little to analyze beyond that it works. The end goal of this project is as a product for users, so it will then be evaluated in a very similar way to the interviews done by Clarke and Duimering, where I will ask users questions encouraging analysis. Intuitiveness of use, appeal, and bugs/glitches are all things to evaluate based on. However, with just the viewing working, all I can evaluate is if the display works as expected and models the way we view things in real life, which it does.

5 Results and Conclusions

The project was successful as far as it has gone. It displays the minimal 3-D world it has, and allows the user to explore it.

Future goals include networking, surface loading, a realistic physics system, and a proximity-based chat system.

Surface loading will allow for actual, recognizable environments for the user to explore. Combined with a physics system, this will allow the user to explore the environment in a way much resembling real life.

The final component, a chat system, would allow users to interact over the network in this virtual world. Furthermore, they would be able to interact through realistic speech, quieted over a distance, rather than a radio or text messaging system, which detracts from the realism in the simulation.

If this is a successful chat system, and users do indeed find it to enhance socialization, and I have every reason to believe they will (Gibbs, Wadley, and Benda), this could be a major point for others build off of. Anyone creating a networked environment for users to interact could employ this system, both for improved realism and greater social enjoyment for users.

6 Acknowledgements

I would like to thank Shane Torbert for introducing me to computer science, as well as teaching me most everything I know about programming, and I'd like to thank Nick Starr and Stephen Drodge for teaching me the rest. I'd also like to thank Randolph Latimer for his support and flexibility with this project throughout the year.

References

[1] Clarke, Delwin, and P. Robert Duimering. "How Computer Gamers Experience the Game Situation: A Behavioral Study." The ACM Digital Library. 2006. 30 Oct. 2008 http://portal.acm.org/citation.cfm?id=1146816.1146827coll=ACMdl=ACMCFID=8522235CFTOKEN=30531755.

[2] Gibbs, Martin, Greg Wadley, and Peter Benda. "Proximity-Based Chat in a First Person Shooter: Using a Novel Voice Communication System for Online Play." The ACM Digital Library. 2006. 30 Oct. 2008 http://portal.acm.org/citation.cfm?id=1231894.1231909coll=ACMdl=ACMCFID=8522235CFTOKEN=30531755.

[3] Norvig, Peter. "Python for Lisp Programmers." Norvig.Com. 30 Oct. 2008 http://norvig.com/python-lisp.html.

[4] Angel, Ed, David Shreiner, and Vicki Shreiner. "An Interactive Introduction to OpenGL Programming." The ACM Digital Library. 2007. 21 Jan. 2009