Exploration of a 3-D Environment

By Zachary Greer

Introduction

Here's what it is.

Abstract

- Allows user to explore 3-dimensional world
- First person
- Mouse and keyboard input

Inspirations

- Proximity-based chat system
  - First employed in first-person shooters
  - Enhanced social element greatly (realism, intuitiveness)
  - Could be used for just a chat client
  - Could also be used for other teamwork activities
- Gaming experience
  - Explains the elements of an enjoyable game
- Google Lively, IMVU are 3-D chat clients, but are neither 3-D nor audio.

Goals

- Realistic physics
- Loadable surface system
- Networked, first-person graphical chat system
- Chat: audio, based on proximity (modeling real life)
- Game element? (Maze? Capture the Flag?)

Development

Here's how it works.

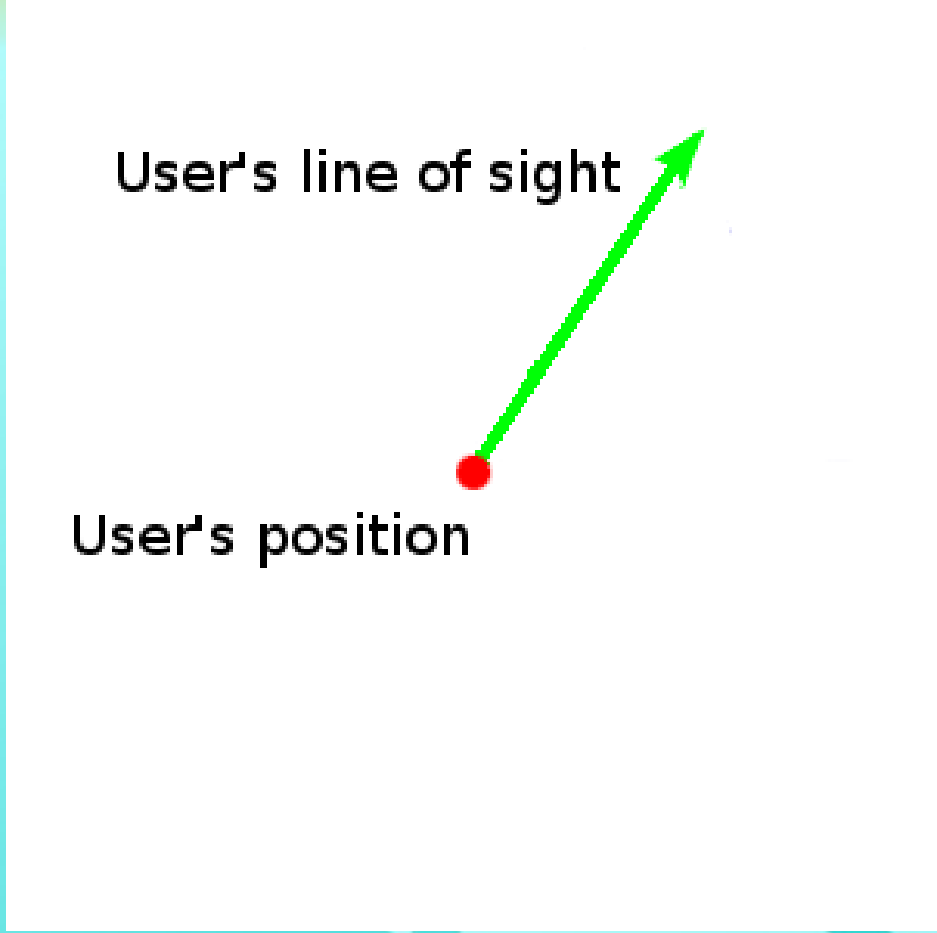
C, OpenGL, MPI

- Why C?
  - C is a powerful and fast language
  - OpenGL is a 3-D graphics library compatible with C
  - C does not change its syntax. C code is always valid
  - MPI is an interface to pass information over a network and is compatible with C
  - I have had experience and am familiar with all of these.

Perspective

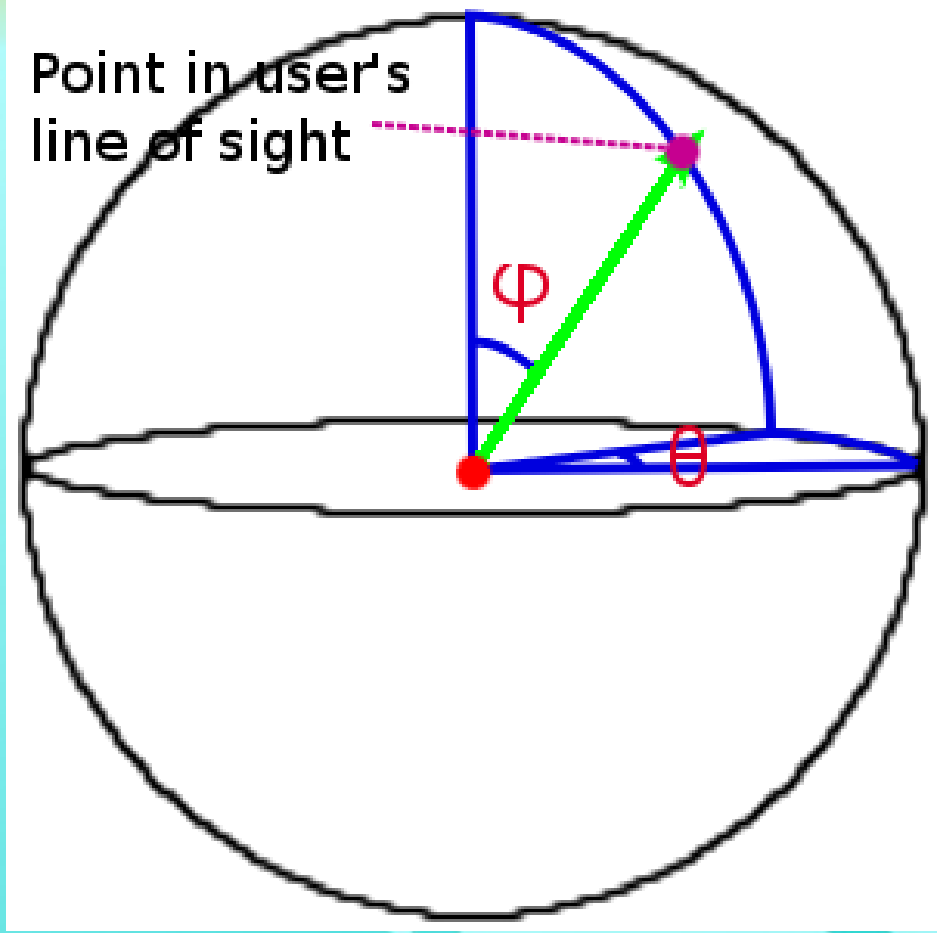
- Spherical coordinate system
- Angle measurements in radians
- Mouse control of looking around

Why Spherical?



- OpenGL provides a command to look at a certain point from another point
- User's position is easy, but how to give point on line of sight?

Why Spherical?



- Apply a spherical coordinate system, and this is simple
- In (x,y,z) format, the point is (x_position_of_player + cos(θ)*cos(φ), y_position_of_player + sin(θ)*cos(φ), z_position_of_player + sin(φ))

Why Spherical?

- Mouse system intuitive, looking up or down changes φ, left and right changes θ
- Movement simple, obtaining a unit vector in the plane of horizontal movement just <cos(θ), sin(θ)>

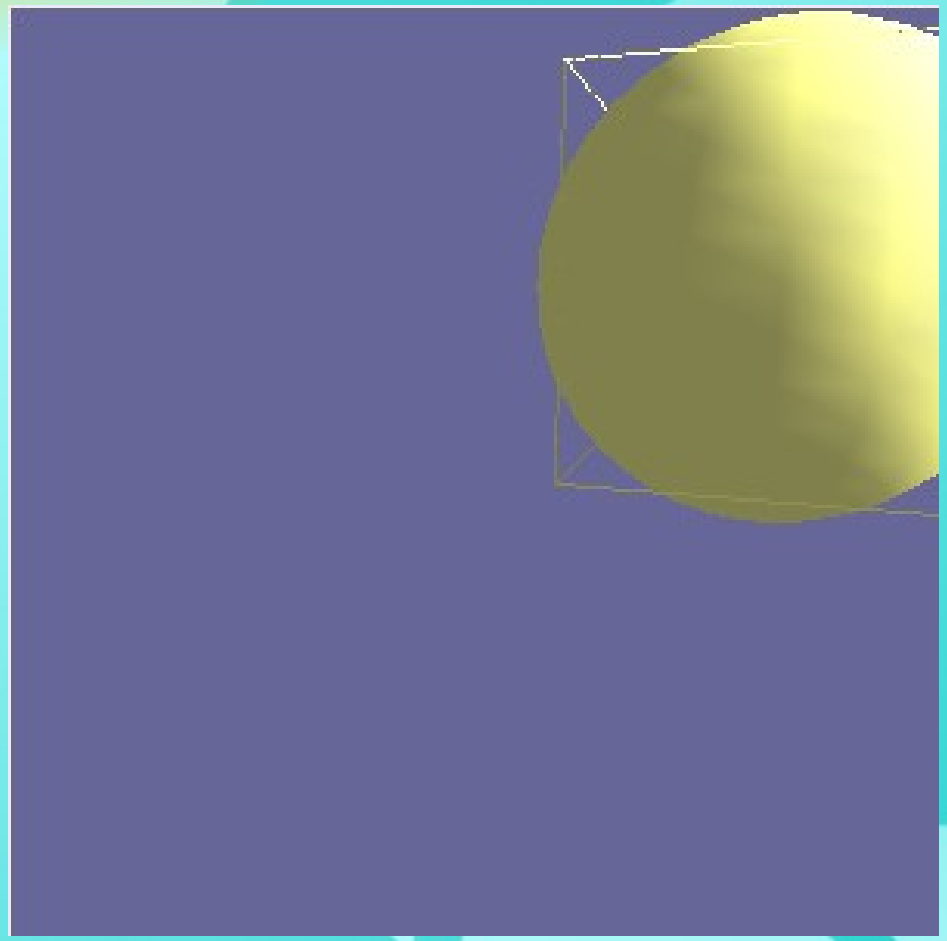
Movement

- Movement occurs with respect to the direction the user is facing
- Movement is controlled with keyboard input
- The unit vector in the direction of the line of sight projected onto the xy plane is multiplied by speed, added to position. The result is motion.

Results

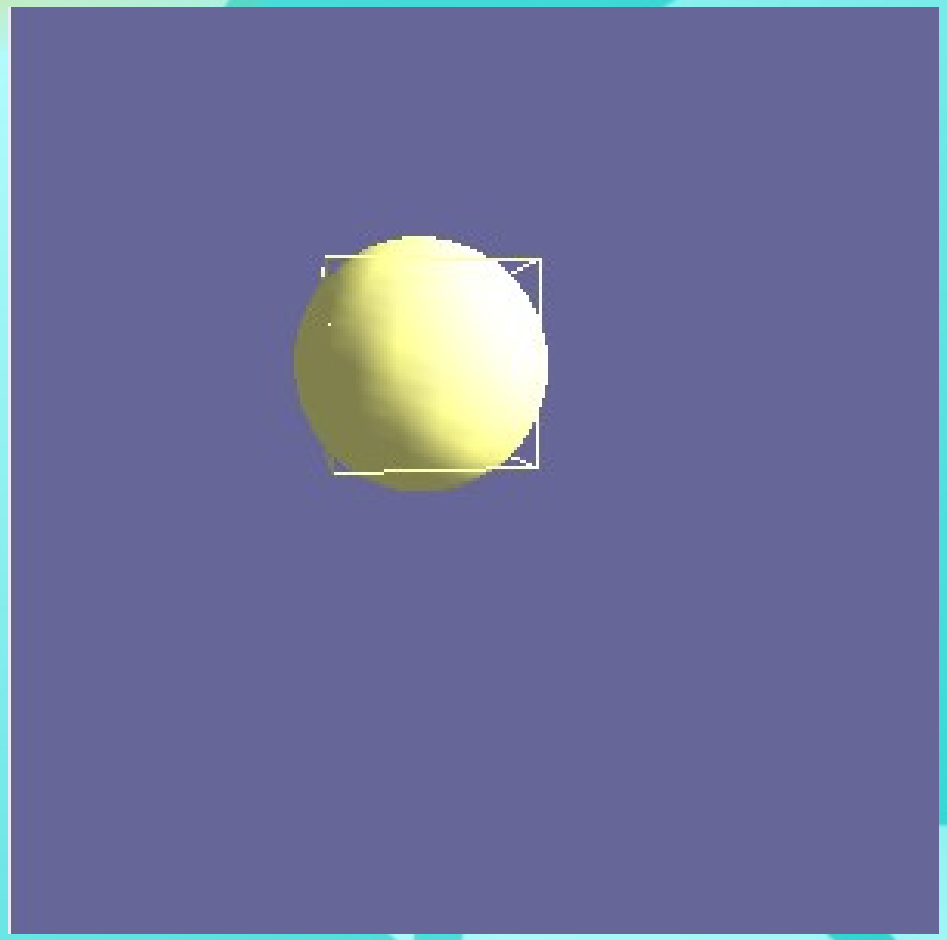
Here's what it does.

Current State



- The user may move along the xy plane
- A sphere and wire cube are provided for reference
- No hit detection is performed

Current State



- The user may move freely throughout the environment with arrow keys
- The user may use the mouse to look wherever they wish

Changes of Plan

- Originally meant to be a simple first-person shooter
- Now the plan is a networked chat system/maybe puzzle game/team activity