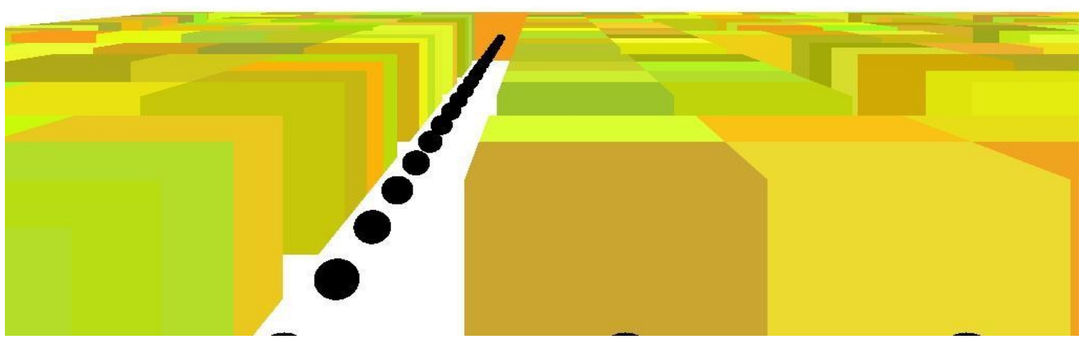
First-Person PacMan

by Brett Jones
TJHSST Computer Systems Lab 2007-2008

Abstract

The purpose of this project is to create a 3D, first-person version of the classic PacMan arcade game in order to learn more about the concepts of 3D graphics programming and rendering algorithms. The project will also include a basic AI to control the ghosts.



An initial view of the scene, slightly raised, with wall colors differentiated, and showing some of the cookies.

Background

Computer graphics is the science of creating virtual representations of objects with computers. 3D computer graphics is a subset of this field, and specifically involves representing three-dimensional scenes with a computer. The field involves modeling (creating the virtual 3D objects), animation (the movement of the models through time), and rendering (projecting the 3D scene onto a 2D screen), as well as mathematics, notably matrix algebra.

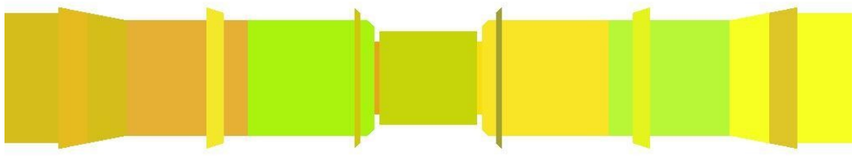
The animation aspect of 3D computer graphics is where much of the math is used, as animations are generally created by applying matrix transformations to a collection of the scene's vertices. Rendering relies more on geometry, both because a 3D scene must be projected onto a 2D screen and because some rendering algorithms depend on or restrict the geometry of the scene. For example, ray casting, a subset of ray tracing, requires right angles - walls, for example, must be perpendicular to the plane of the eye. Ray tracing does not have this restriction, nor does polygon modeling (also known as rasterization). These two algorithms create realistic representations of scenes; ray tracing is more so (in both the method of implementation as well as the resulting image), but it is considerably more computationally expensive and is thus uncommon for realtime rendering. This project will use the native Java3D renderer, which uses a rasterization algorithm.

Progress

The program is coded to run in fullscreen exclusive mode (FSEM) in order to display the game over the entire screen. The program runs without errors and displays the scene objects, and the view can be rotated. The menu consists of a title image and seven function buttons: New Game, Control, Sound, Save Game, Load Game, High Scores, and Quit. Quit exits the program, New Game creates an instance of the World class (which extends Frame) and sets the program to run in FSEM with the World class as the viewable display, and the other buttons do not have any coded functionality.

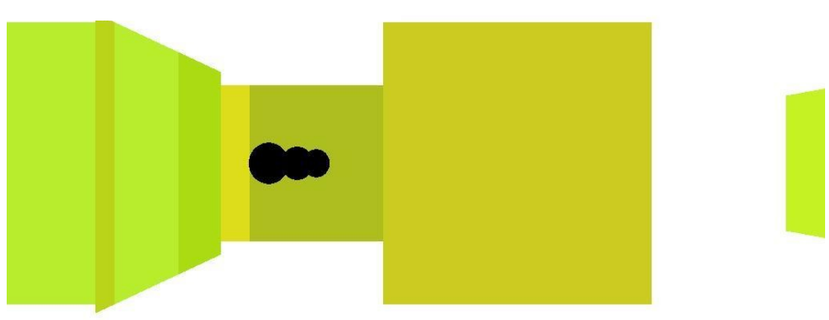
Inside the game, the program displays a black background with randomly shaded blue cubes (the wall objects) connected in the fashion of contiguous walls, white spheres for the cookies with larger white spheres for the fruits, and accepts keyboard input for motion and returning to the main menu (Figure 2). The program allows motion throughout the scene, but does not allow interaction with the scene; thus, the player can move through the walls and cannot eat the cookies or fruits.

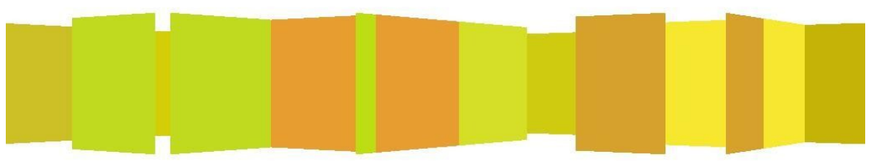
The program uses keyboard input to control the motion through the scene. The right and left arrow keys turn the viewer through a right angle to the right and left respectively, while the up arrow sets the player to forward motion and the down arrow turns the player around to face the opposite direction. The keys do not need to be held down to work; a single tap of the appropriate key will cause the program to perform the appropriate function automatically. However, the forward movement is the only function that performs continuously - the turning functions perform once per keystroke. In addition, each key can interrupt whatever function is currently being performed. The program also listens for the P and Escape keys, which pause the game and end it, respectively.



A screenshot of the scene's initial view, as of 04/02/08.

A screenshot showing the cookies in the scene.





A screenshot showing a slanted view of the scene after using the encoded rotational animation.

The images on this poster have inverted colors.