

TJHSST Computer Systems Lab Senior Research Project Virtual PacMan 2007-2008

Brett Jones

May 8, 2008

Abstract

The field of 3D computer graphics has been explored quite extensively, and comprises of three major parts: 3D modeling, animation, and 3D rendering. The first part, 3D modeling, refers to creating a 3D representation of an object. Animation, the second part, is moving the object through time. The final part, 3D rendering, is drawing the animated 3D model to the screen. 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, and the 3D rendering will be left to the native Java3D renderer.

: **Keywords:** 3D graphics, rendering algorithms, AI

1 Introduction

1.1 Scope of Study

The program, for the purposes of this project, will only consist of the core of the game; i.e., the program will only have the visual display (with control inputs) and a basic AI for controlling the ghosts. The functions such as save/load game, sound, customizing controls, high scores, etc. will be absent

from the project unless time permits the inclusion of these features; the 3D graphics is the main focus of the project and thus has top priority.

1.2 Type of Research

This project is pure applied research.

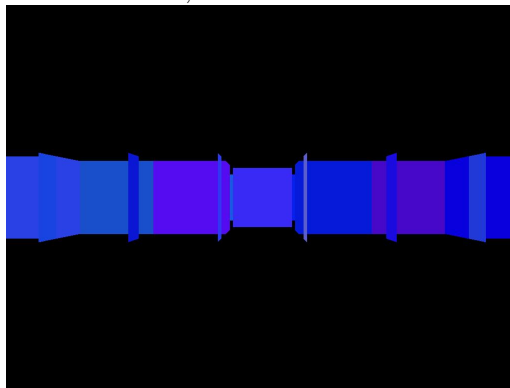
2 Background

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 very realistic representations of scenes; ray tracing is more realistic (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.

3 Progress

Currently, 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

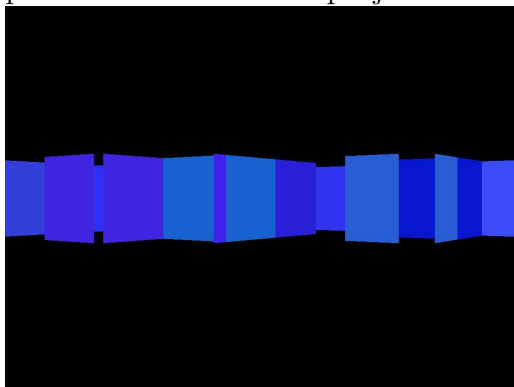
Frame) and sets the program to run in FSEM with the World class as the viewable display, and the other buttons do not have any coded functionality. The program displays a black background with randomly shaded blue cubes (the wall objects) connected in the fashion of contiguous walls, and accepts keyboard input for motion and returning to the main menu. The move method currently generates runtime errors, but the turnLeft and turnRight



methods work appropriately.

4 Procedures and Methodology

The program will be coded with Java and Java3D using the jGRASP compiler. The main focus of the project will be the 3D graphics portion, which hopefully can be completed with the ray tracing algorithm. Afterwards, coding the AI for the ghosts will commence, and once the AI is complete, most of the remaining programming time will be devoted to optimization/debugging, with the addition of the extra features of a game mentioned above if time permits. Visuals of the project will consist primarily of in-game screenshots.



The best test of the program is playing the game. Bugs that manifest

themselves in the visible part of the program (such as ghosts moving through walls, walls in the wrong place, etc.) will become apparent by playing the game. Playing the game also provides a general idea for the runtime speed as well, as the game will be noticeably laggy if the code isn't efficient enough. Another method for testing the efficiency of the code is the "time" function of the Linux Bash terminal. This function accepts a command as an argument (in this case, a call to the Java Virtual Machine to run the program) and reports the total, user, CPU, and system times spent on the program. The CPU time represents the processing time of the program, a key indicator of efficiency (high numbers here are not good); the user time represents how long the program was waiting for user input; the system time represents how long the computer spent waiting to run the program (a number that should only be significant when many other programs are running on the same system); and the total time is the sum of the user, system, and CPU times.

4.1 System Requirements:

Operating System: Windows, Linux, or Solaris

CPU: TBA

RAM: TBA

Graphics Card: OpenGL 2.0 compatible graphics renderer

HDD Space: TBA

4.2 References:

<http://www.vrupl.evl.uic.edu/LabAccidents/java3d/lesson06/indexb.html>

-A Java3D tutorial dealing with the Appearance object.

<http://benmoxon.info/Java3d/index.htm>

-A general Java3D tutorial.

http://www.cs.mu.oz.au/380/project/Patricks_java3d_tute/tutorial.html

-Another general Java3D tutorial.

<http://www.few.vu.nl/~kielmann/theses/avdploeg.pdf>

-A research paper about the viability of raytracing in realtime programming.

<http://graphics.cs.uni-sb.de/~jofis/Arbeit/GI04-MWZC-RayTracing.pdf>

-A research paper about raytracing, with real-world examples of the algorithm using Quake 3 and a program built from scratch.

<http://graphics.cs.brown.edu/games/quake/quake3.html>

-A description of how iD Software, Inc. coded the BSP file format for Quake 3 maps.