Design of a 3D Graphics Engine
TJHSST Senior Research Project Proposal
Computer Systems Lab 2009-2010

Joseph Hallahan, period 5

January 25, 2010

Abstract

One important idea important to many modern programming problems is separating source code into multiple parts. Although it makes the program more complex, code from one part of the program can be able to be used for other programs as well. The only difficulty there lies in keeping track of the structure. In Java, different classes can extend each other and implement interfaces. In C++, classes can still extend others and header files, which are basically just a list of declarations that can be used in any program, can be included. This sort of organization seems to be a good way of creating a graphics engine, since any complicated rendering methods can be placed into separate files and can therefore be used in any number of programs. This project involves creating a graphics engine in this fashion that, in addition to rendering, can also implement different forms of collision detection and keep track of input. C++ and OpenGL will be used to create this engine.

keywords: graphics engine, OpenGL, C++

1 Introduction

1.1 Situation/Statement of Problem

Engines are parts of programs that provide functionality for certain processes. Some engines focus on the computations needed for calculating realistic physics simulations, while others facilitate the rendering of graphics. One feature common to both physics and graphics engines is the fact that both are typically not limited to one program; often, engines are used for several applications. Programmers can have the choice whether to create a program entirely from scratch or to build it off of an existing engine.

1.2 Purpose

The goal of this project is to create an engine that focuses predominantly on graphics. It will be able to be used for many applications, such as for simulations, games, and other 2D and 3D applications. It will need to have the ability to do the math related to these processes, as well as the implementation of various collision detection techniques. The engine will also have some ability to render text and to get input from the mouse and keyboard.

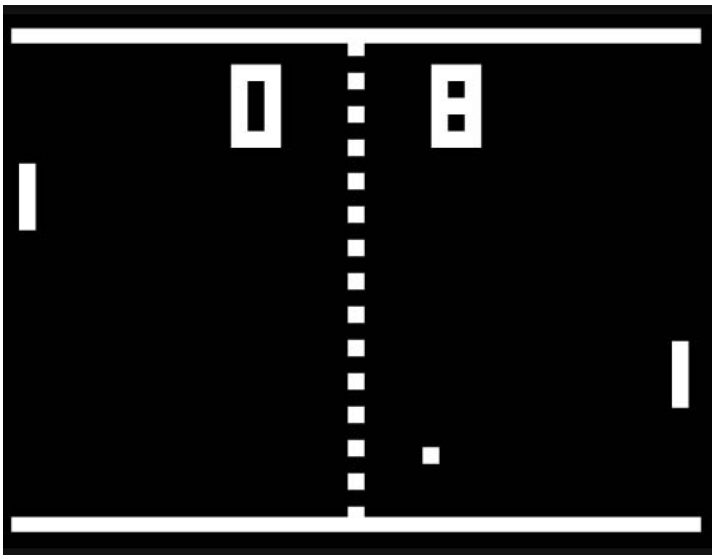

Figure 1: This is the ultimate goal of the project: to create Pong without using graphics code.

2 Background

Graphics engines are very important to the work done by many programmers today. In the process of creating an application, after the initial design phases are complete, the first thing the programming team will typically do is begin work on the application's engine. Some applications, if they are new and different from what has been done before, will need an entirely new engine and, as a result, will require a lot more work. However, if it is similar to one that has already been completed and released, it is much easier to use an existing engine than to create a new one. In addition to reqiring less work, completed engines have already been tested for stability and compatibility, so programs that use them typically do not need as much post-production work.

Because graphics code can become rather complex and not user-friendly, it would make an excellent candidate for use in an engine. Encapsulating all the code needed for rendering graphics would take a large burden off of programmers who will then be able to spend more time on the less rigid parts of projects. The basic code for graphics does not change very much from application to application, so putting it in a separate file for other programs' use would be a good idea.

3 Development

3.1 Structure in C++

Last quarter, I explored the use of the OpenGL library in conjunction with a C++ source file. I created a program which could create images and then draw them. One of these examples featured user input. This

quarter, I improved upon this by creating my own header file and including that into a Pong program. The purpose of this header file was for the program including it to be able to use the OpenGL methods (from the header file) without needing any explicit OpenGL code. I wrote methods that could draw simple shapes, such as circles and rectangles, as well as text. Technically, this code was not in the header (.h) file, but with a separate source (.cpp) file with the same name. The header provided the declarations for these methods, and those were what the Pong program needed to compile and run correctly. It was able to link with the implementations in the separate source file and use them.

3.2 Collisions

Collisions are an important area in both games and simulations, and as a result they are a necessity in nearly every sort of engine, even one otherwise exclusively used for rendering. As the engine in this project already supports more features than the average graphics engine, it makes sense that collision detection be added as well. Right now, only one form of collision detection is present in the program. This type, in which objects are checked for collision on a frame-by-frame basis, is called discrete collision detection. This type is useful in simple applications since it is fast and easy to code. All that is necessary for discrete collision detection is the comparison of the pixels covered by each shape, to see if any of them overlap. There are a number of modifications that I could make to the engine involving collision detection in the future. Continuous collision detection is a process which checks to see if two shapes have moved through each other, an issue that can occur in discrete collision detection if care is not taken. This would be a simple feature to add. Other collision detection techniques, such as those used for concave polygons, would be useful as well in order to add as much functionality as possible to the engine. However, the discrete collision detection I have now is more than sufficient for the simple game of Pong used to show the results of the current engine. Whichever collision technique is needed will depend on the scope of the project at hand.

4 Tests and Analysis

After I finished my header file, my next step was to get it to be compatible with a simple game of Pong I had coded a little while earlier. Pong programs are not new to me; I have created them with Java and C++ before, and hopefully the header file's OpenGL methods would make this program work as well. Unfortunately, I had numerous problems getting them to work together. Sometimes one piece of code wouldn't be able to see any of the methods from another part, and sometimes simple syntax errors slowed progress down. In the end, I resorted to another IDE called Code::Blocks, and that the problem of linking different source files together. After fixing a few more errors, the code was able to compile and run, and a little debugging resulted in a functioning game of Pong.

5 Results

I was able to successfully implement the methods of my OpenGL engine into a basic game of Pong. The user can input a direction using the keyboard and move a white rectangle up and down. When the white circle hits the rectangle, the engine identifies it as a collision and reverses the circle's horizontal velocity. The engine also displays the scores at the top of the screen. This program was a good test of my engine, since it used every method I have coded so far, and the actual source of the game (Pong.cpp) includes little to no actual OpenGL code at all - it was successfully encapsulated within the engine.

6 Conclusion

This program, when linked to the header file containing the OpenGL methods, correctly used them despite not using any explicit OpenGL code within its own source. Relating to my original goals for this project, it was a success. However, it needs to be able to do significantly more work in order to be truly considered an engine. This current project can, at best, be considered only a library, which declares and implements complicated methods once so they do not have to be rewritten for every program that uses them. While I was working on this, I was more worried about encapsulating the OpenGL code since it is so complicated, but now that it works correctly I can start to be less specific about what the engine needs to do. In other words, OpenGL programs share significantly more code than just the graphics code. Any program has certain parts, such as the initializion of variables, the code needed to end it, and, most importantly, a loop which continuously does calculations and then draws objects on the screen. This is much more work than a simple header file would be able to accomplish, so my next step on this project will be to create a dedicated engine class that will have so much built-in functionality that it will be able to run itself with no added work from a programmer.

7 Bibliography

- Development of a 3D Graphics Engine (Kassing)
- Modular Architecture for Computer Game Design (McNeill)
- Multi-threaded Game Engine (Tulip, Bekkema, Nesbitt)
- Interactive 3D Geometry in OpenGL (Welsh)
- FROG: The Fast & Realistic OPENGL Displayer
- Some code used from "http://www.tjhsst.edu/ dhyatt/superap/opengl.html"