Design of a 3D Graphics Engine for use in Various Applications

Joseph Hallahan
Computer Systems Lab 2009-2010

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. In C++, classes can 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.

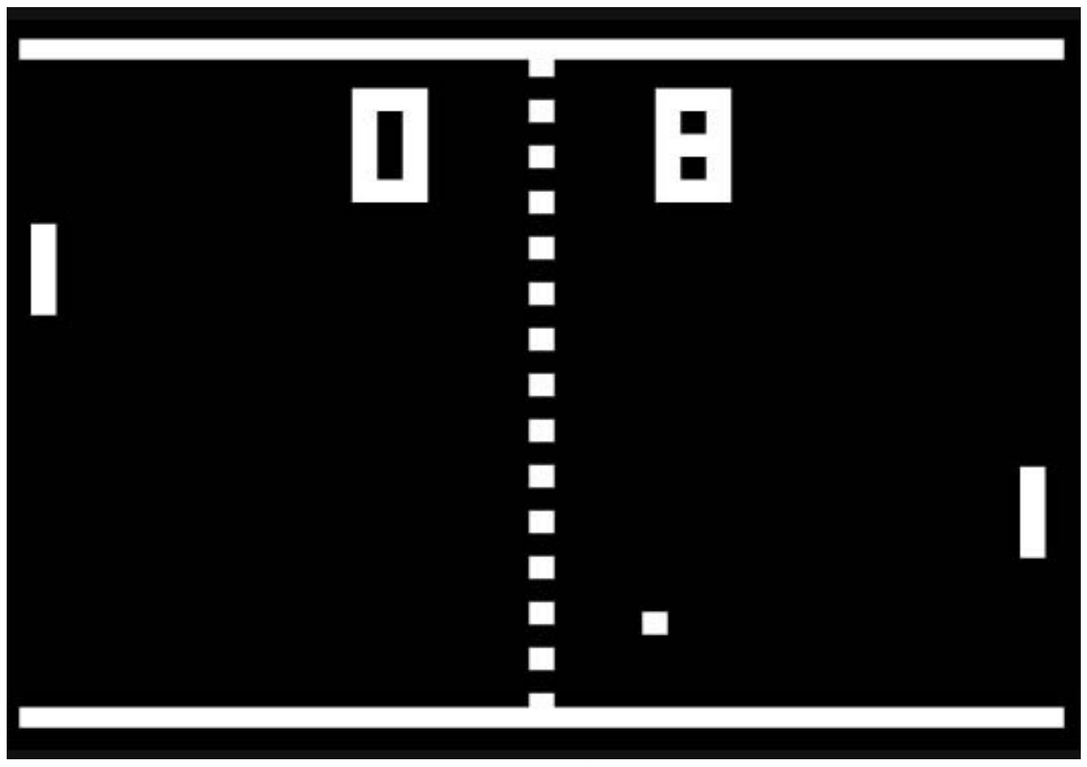

Fig 1: The plan...

Background and Introduction

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. 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.

Discussion

Unfortunately, while the program ran without issues, I neglected to provide the code needed to create a window in which to draw the game. I realized this too late and when I found out, I put some of the code from my 1st quarter project into my current program, which was my purpose for creating that 1st quarter project in the first place. However, the computer I was using did not have the OpenGL source I needed to create the window, and when I tried it on the school computer, it wouldn't link the files together. However, using a simple form of text output, I was able to ensure that the program still worked as it was supposed to. Next quarter I will not use the same structure as this one so that I will not have these OpenGL or linking problems.

Results and Conclusions

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. 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.