The Reusability and Portability of Project Graturmach

Much of my experience in coding comes from working in server-side website coding languages. Over the years, I have formed my own opinions on how important code reusability is to software development. I have seen firsthand how much time can be saved if you do not have to type out every line of code anew every time you start a new project. This concept is especially relevant in web development; for instance, I have reused large amounts of code at least three distinct times from a blog entry and comment backend that I built years ago, vastly reducing the amount of time and effort needed to implement those websites. Portability is also an important part of software development, as it is vital to know your market and provide a product that works well on whatever platform they are using. Simply put, you will not make money if your product runs only in Windows and 100% of your potential customers use Linux.

Luckily, there are times when making money is not the foremost imperative. Many research projects have no need to pay attention to their “market,” since they really do not have one in the conventional money-making sense. Project Graturmach falls into this category: it was never meant to be sold, and therefore it is up to the software engineers to determine what platforms it runs on. That is just portability of software, however, and not portability of code. While Project Graturmach is not currently able to be compiled in Windows, the code it uses is very portable. The OpenGL is a cross-platform graphics standard that is known for its interoperability with different systems – OpenGL programs will look the same on any system that has implemented the standard correctly. The user interface that Project Graturmach uses is also very portable. It uses only OpenGL to render the user controls, and by doing so, they appear
the same on whatever platform the program is running on, which is a claim that is somewhat of a rarity in the realm of software user interfaces. Otherwise, the only other piece of code in all of Project Graturmach that is system-specific is the timer delay function; this function pauses the system for a number of milliseconds, and must be platform-dependent due to different handling of clock delays in the hardware of each platform. Other than the delay function, there is nothing preventing Project Graturmach from being ported to any system that simply implements both a C++ compiler and supports the OpenGL specification within C++.

The reusability of Project Graturmach is another story. Graphics applications, especially ones with user interfaces, are more complicated than comparable backend-only applications, and have many attributes that make code reuse difficult. Much of the graphics commands are hardcoded to values, and many graphics functions are built only to serve one particular and non-abstract purpose. Taken as a whole, Project Graturmach would be easily reused if the goal was to add a Turing Machine simulator to an already planned software project, mostly due to the encapsulation that the TMGraphics class provides. But separated into discrete chunks, the code components of Project Graturmach are not overly useful outside of the context of a Turing Machine program. Much of the problem with the lack of reusability lies with having to conform to the OpenGL specification, though this problem would not be specific to OpenGL. Graphics APIs are simply too complicated to allow very seamless code reuse; one graphics system, like OpenGL, may go about its graphics rendering entirely differently than another, like DirectX.

Graturmach’s source code is quite portable, and only relies on a few givens – OpenGL and C++ provided – to assure a faithful reproduction of results, with one system-specific exception, that of the system delay. Graturmach’s source code is not very reusable if not taken as a whole, but most of the difficulty is due to the fact that the project is light on backend non-
graphical components and heavy on front-end API-dependent display components. Overall, I am satisfied with this mix of attributes, and I feel its code portability and reusability is as good as could be expected of a project of its class.