Prof. Joel Adams (advisor)
Christiaan Hazlett
Elizabeth Koning

Thread Safe Graphics Library, Summer 2017

One fundamental problem which computer scientists often encounter is that of allocating scarce or shared resources between concurrently running applications. In many modern programs, multiple bits of code, called threads, need to access or modify the same data store, but may run into issues when one thread attempts to read while another is writing to the data store, or both attempt to write data at the same time. In other cases, multiple programs may need to share access to scarce resources, such as a printer or a screen. In order to remediate these problems and avoid causing a system to become unstable, threads must cooperate with one another.

Computing textbooks have included several now-classic examples of these problems for years, but they have been only theoretical, and there has never been any way to easily visualize them before Calvin's Thread-Safe Graphics Library. TSGL makes it possible to create programs that visually demonstrate real implementations of these problems and their solutions, which helps students intuitively grasp what can otherwise be extremely difficult concepts.

This summer, we enhanced the existing visualizations that came with TSGL, as well as created new ones for three of the most common textbook multiprocessor synchronization examples—the Reader/Writer Problem, the Producer/Consumer problem, and the Dining Philosophers example. In order to create these visualizations, and in order to target a wider range of hardware than we supported before, we needed to start from scratch and re-write most of the library, as well as add a significant number of new features.

To date, we have re-engineered the underlying graphics system of the library in order to be compatible with older and lower-cost hardware that is more accessible to schools, in particular the Raspberry Pi. We have also created a new programming interface for the library which makes TSGL much easier to use for the end-user, adding layers, following object-oriented programming paradigms and allowing addition, removal, and mutation of existing object on TSGL's canvas drawing context, all while maintaining backwards-compatibility with existing applications of the library.

Personally, the opportunity to work on TSGL over the last several months has been a great chance to grow my knowledge of the C++ programming language and the management of concurrent threads and scarce resources in a software system. It has also given me extensive knowledge of how the graphics subsystem on modern computers works, and how to create software that talks to the graphics card in order to draw pictures on-screen. Finally, this research project has helped me learn how to effectively work with a project partner to satisfy the needs of real users in the education field.