Similar Projects

During our external research, we could not find another project where someone implemented a graphics library inside of Coq, however, there are many other functional languages that implement a graphics library. Examples of others are: Haskell, Scheme, Ocaml and pretty much any other functional language in which we can use axioms to instantiate the graphics primitives. The functional language that we focused on for our implementation of a graphics library was Ocaml graphics library. We chose to use the Ocaml graphics library as the model for our Coq graphics library for a few reasons. The most important aspect of Ocaml that led to use using it over any other functional language is that Coq is written in Ocaml, which makes translating and exporting functions from Ocaml to Coq more intuitive than if we had used another language like Haskell or Scheme. We took functions from the Ocaml graphics library like plot, open\_graph, color, lineto and implemented equivalent functions in Coq and then exported them as Ocaml to be displayed. In addition to recreating some of the Ocaml graphics functions in Coq, we also implemented some supporting functions like draw\_rect and fill\_rect and we proved some theorems about the specifications of these functions.

Even though there have not been any attempts to implement a graphics library in coq, there has been research done in the verification of the specifications of geometric shapes in Coq. In Pham and Bertot’s paper “A Combination of a Dynamic Geometry Software With a Proof Assistant for Interactive Formal Proofs,” \cite{geo} they adapted Coq’s tactic language to allow users to interactively construct simple proofs about geometric shapes.

Conclusion

In this paper, we outline the creation of our implementation of a graphics library in Coq. In this project, we were able to implement all the parts that we expected to, but there is still a lot more work that we can do to add more functionality to the project.