Homework 1 – Graphics Warmup

Due: September 19th (11:59pm)

Overview: In this homework assignment you'll implement several basic functions relating to the 2D planar geometry of points, lines, and triangles (e.g., computing distances and intersections).

Description: You must implement a 2D geometric library by completing the implementation of the 18 functions in *geom_lib_2d.h*. These functions move points, intersect lines, calculate distances, compute areas of triangles, and more. You will only turn in the single file *geom_lib_2d.h*. Please make sure your implementation is either completely self-contained, or makes use of only the unmodified functionality present in the included PGA library (*pga.h*, *primitives.h*, and *multivector.h*). You may not make use of any other external code.

The intention of the assignment is both to use concepts in projective geometric algebra and to learn to use the PGA library files we have provided you. While the library is not required, using it means many of the problems can be solved in just one or two simple lines of code. You are free to use other (non-PGA) approaches as well, as long as you consistently get the correct answer for all tested inputs.

You are strongly encouraged to look at the sample file *pga_example.cpp*. This file showcases several of the mathematical operations supported by the PGA library (wedge, vee, dot, normalize, …), with a special focus on those aspects of PGA needed to complete the assignment.

Submission Details:
You will turn in only the file *geom_lib_2d.h*. Additionally, your code must be able compile using the following command on the lab machines:

```
g++ -fsanitize=address -g -std=c++14 main.cpp -o geom_test
```

If you have not used `-fsanitize=address` before, it's very useful. This command tells the compiled code to stop execution and report out to the console anytime the program reads or writes from memory it didn't create. This feature is invaluable for catching subtle memory access errors (such as writing past the end of an array).

The `-g` option tells the compiler to save useful information for debugging. The `-std=c++14` option says to use the 2014 variant of C++ (rather than the default 1998 version). C++ underwent a major revision in 2011, and using the 2014 version ensures people are able to use modern features if they choose.