24780 Engineering Computation: Problem Set 6

(*) In the following instructions (and in all course materials), substitute your Andrew ID wherever you see *yourAndrewId*.

You need to create a ZIP file (which may appear as a compressed folder in Windows) and submit the ZIP file via the 24-780 Canvas. The filename of the ZIP file must be:

```
PS06-YourAndrewID.zip
```

For example, if your Andrew account is hummingbird@andrew.cmu.edu, the filename must be:

```
PS06-hummingbird.zip
```

Failure to comply with this naming rule will result in an automatic 5% deduction from this assignment's credit. If we cannot identify the submitter of the file, an additional 5% credit will be lost. If we are ultimately unable to connect you with the submitted ZIP file, you will receive 0 points for this assignment. Therefore, ensure strict adherence to this naming rule before submitting a file.

The ZIP file must be submitted to the 24-780 Canvas. If you find a mistake in a previous submission, you can re-submit the ZIP file with no penalty as long as it's before the submission deadline.

Your Zip file should contain only two files, ps6.cpp. Do not include project files and intermediate files generated by the compiler. But, do not worry about some files or directories that are automatically added by the archiver (__MACOSX__ file for example).

Notice: The grade will be assigned to the final submission only. In the case of multiple file submissions, earlier versions will be discarded. Therefore, when resubmitting a ZIP file, it MUST include all the required files. Also, if your final version is submitted after the submission deadline, the late-submission policy will be applied, regardless of how early your earlier version was submitted.

Ensure that your program can be compiled without errors on one of the compiler servers. Do not wait until the last minute, as the compiler servers may become very busy just minutes before the submission deadline!

Submission Due: Please refer to Canvas.

START EARLY!

Unless you are a good programmer, there is no way to finish the assignment overnight.

Equation and SimultaneousEquation classes [ps6.cpp] (100 pts)

You can start from the base code, or you can start from scratch. The base code is a modified-version solution to Problem Set 1.

Write a C++ program that solves linear simultaneous equations:

$$ax + by = c$$

$$dx + ey = f$$

The solution to the equations is:

$$x = (ec - bf)/(ae - bd)$$

$$y = (af - cd)/(ae - bd)$$

The program should prompt the user to enter a,b,c,d,e, and f as:

```
ax+by=c
dx+ey=f
Enter a b c d e f:
```

And take these parameters from the terminal. Then, if the absolute value of $ae - bd$ is less than the variable called *tolerance*, which is defined as:

```
const double tolerance=1e-6;
```

then print:

```
No solution.
```

Otherwise, print the solution like:

```
x=? y=?
```

Your program needs to have two classes, *Equation*, and *SimultaneousEquation*. *Equation* class must remember three coefficients (a,b, and c) of an equation. It also must have a function called *Plot*, which takes three floating-point values R, G, and B. *Plot* function must draw a straight line defined by the equation with the specified color (specified by R,G, and B), in 600x600-pixel window. Use the scaling factor of 10, i.e., the X value at the left and right edges are -30 and +30, Y value at the bottom and top edges are -30 and +30, respectively. Make sure your *Plot* function does not divide anything by zero, and it should be able to plot correctly for any combination of a and b.

SimultaneousEquation class has two *Equation*s, and a member function called *Solve*. This function takes two references to doubles, &x and &y, and returns *bool*. If there is a solution, it writes the solution in x and y and returns *true*. If there is no solution, it does not touch x and y, and returns $false$. This class also has a member function called *Plot*, which takes no parameter. It needs to call *Plot* function of two *Equation*s. The first one needs to be red, and the second one needs to be blue.

Your main function look like the following:

```
int main(void)
{
    SimultaneousEquation eqn;
    double x,y;

    std::cout << "ax+by=c\n";
    std::cout << "dx+ey=f\n";
    std::cout << "Enter a b c d e f:";

    std::cin >>
        eqn.eqn[0].a >> eqn.eqn[0].b >> eqn.eqn[0].c >>
        eqn.eqn[1].a >> eqn.eqn[1].b >> eqn.eqn[1].c;

    if(true==eqn.Solve(x,y))
    {
        std::cout << "x=" << x << " y=" << y << '\n';
    }
    else
    {
        std::cout << "No solution.\n";
    }

    FsOpenWindow(0,0,600,600,1);
    for(;;)
    {
        FsPollDevice();
        if(FSKEY_ESC==FsInkey())
        {
            break;
        }
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glColor3f(0,0,0);
        glBegin(GL_LINES);
        glVertex2i(300,0);
        glVertex2i(300,600);
        glVertex2i(0,300);
        glVertex2i(600,300);
        glEnd();
        eqn.Plot();
```

```
        FsSwapBuffers();
    }
    return 0;
}
```

3 Extra Points: Add a class called *Axes*, which has one member function *Draw*. This function should draw a grid line instead of just X- and Y- axes. X- and Y- axis must be black, and the rest gray (R,G,B)=(0.7,0.7,0.7). Replace lines in the main function appropriately to draw grid lines with this class.

Note that if you end up making an error in an attempt to get this extra points, you may rather lose points.

Sample Console Input and Output

```
ax+by=c
dx+ey=f
Enter a b c d e f:1 1 0 -1 1 4
x=-2 y=2
```

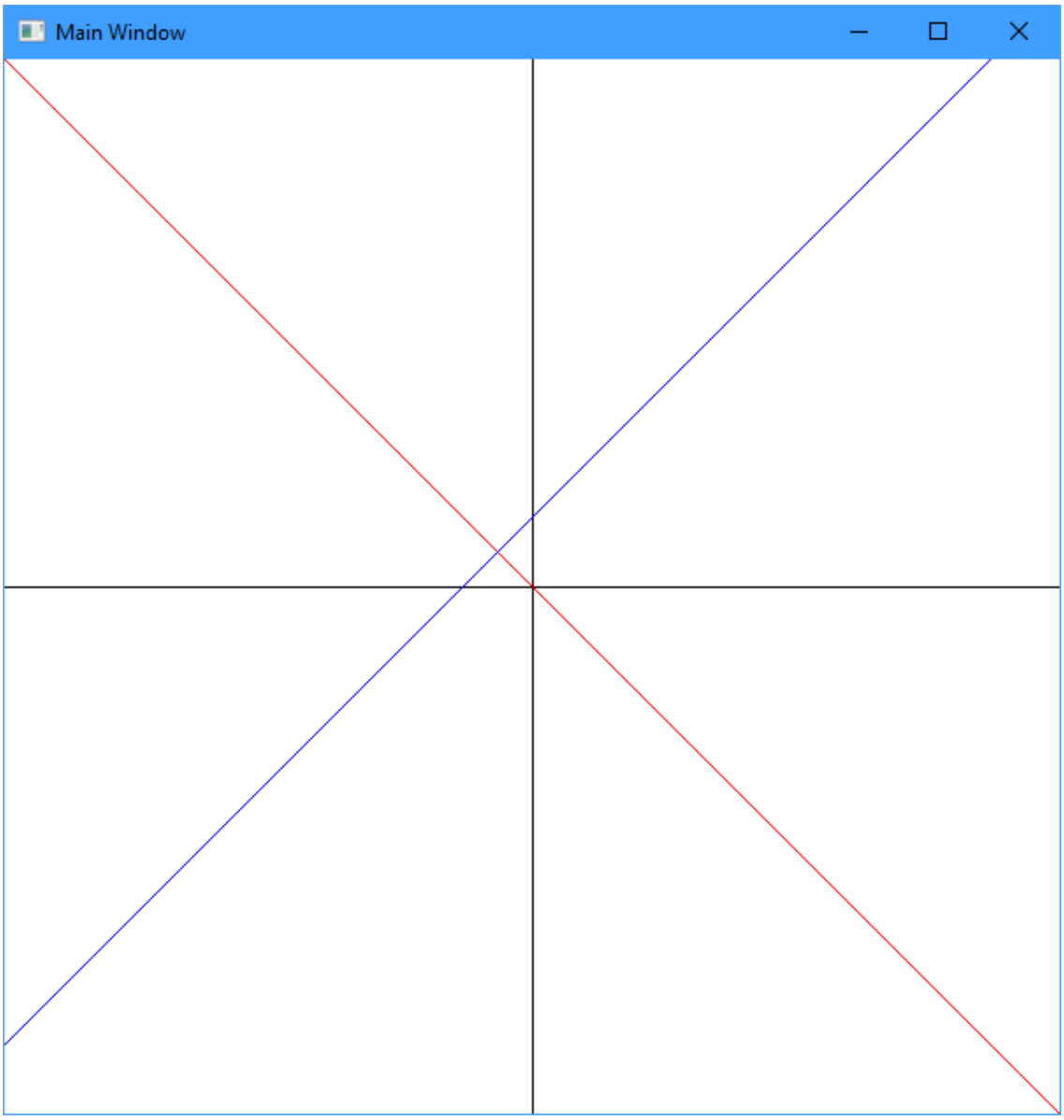

Fig. 1: Screenshot of the plot

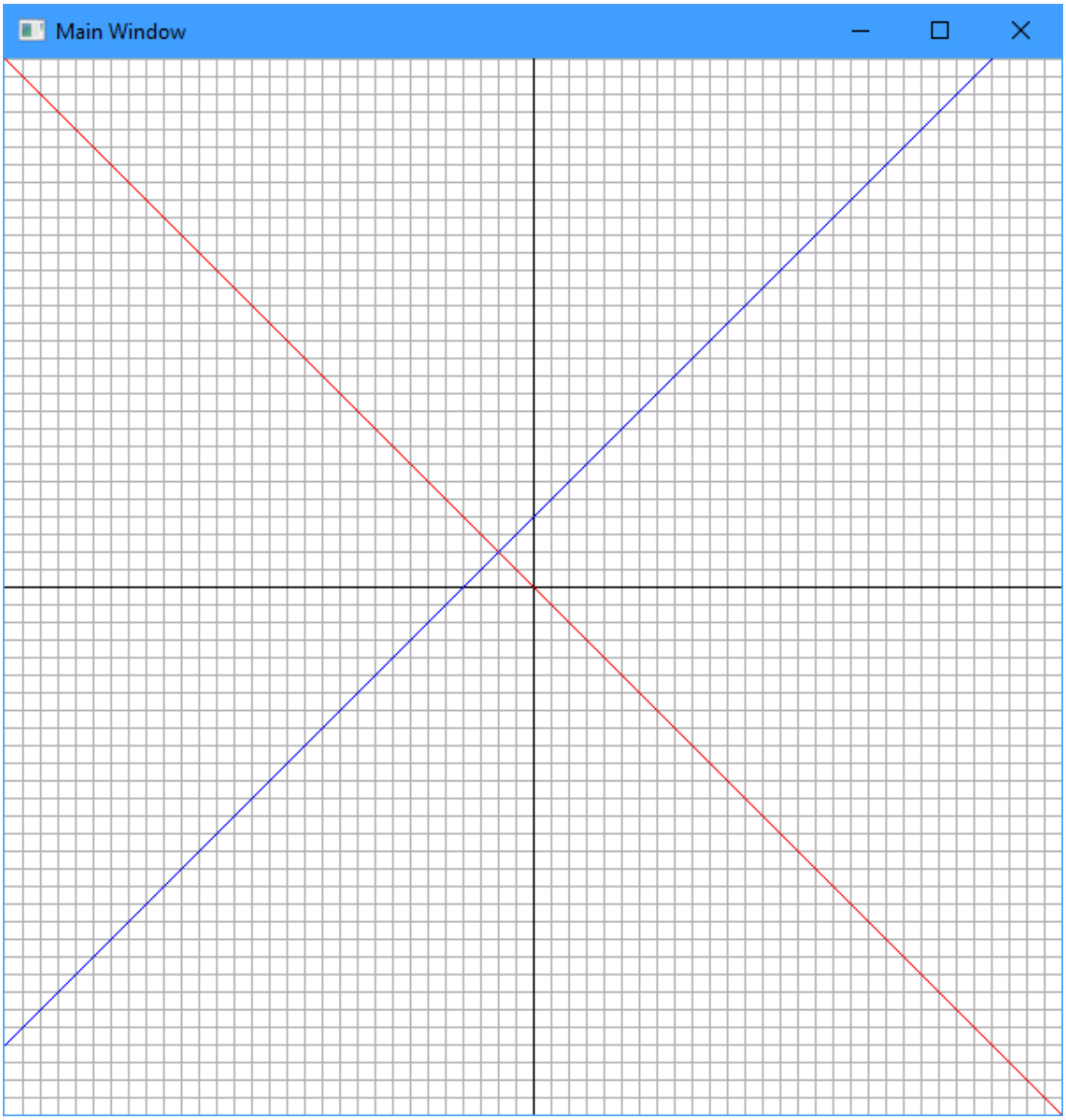

Fig. 2: Screenshot of the plot (for Extra points)

Test Your Program with One of the Compiler Servers

Test your program with one of the following compiler servers:

```
http://freefood1.lan.local.cmu.edu
http://freefood2.lan.local.cmu.edu
http://freefood3.lan.local.cmu.edu
http://freefood4.lan.local.cmu.edu
```

You need to make sure you are not getting any errors (red lines) from the compiler server.

It is a good practice to remove warnings as well. However, we will not take points off for warnings as long as your program satisfies requirements of the assignment.

You can only access these servers from CMU network. If you need to access from your home, use CMU VPN. Please visit the CMU computing services web site how to install the VPN.