Circle Games

Due: January 23, 2004, 5:00 PM
50 points

1 Archeology

The Greek city of Corinth contains a stadium enclosing several race tracks. In 1980 a curved starting line for one of the tracks, shown in Fig. 1, was excavated by the American School of Classical Studies at Athens. This starting line, dating from about 500 B.C., appears to lie on a large circle. By fitting a circle through this starting, line archeologists were able to discover that the starting blocks were 1° apart, strongly suggesting that in 500 B.C. the Greeks used degrees as a unit of angle.

Figure 1: The starting line on a race track at Corinth.

In this assignment you will create and implement an algorithm that performs a part of the circle fitting procedure first used by the archeologists.[1]

[1] A more complete description of the problem can be found in Chris Rorres & David Gilman Romano, "Finding the Center of a Circular Starting Line in An Ancient Greek Stadium," SIAM Review, **3**9, pp. 745-754 (1997).

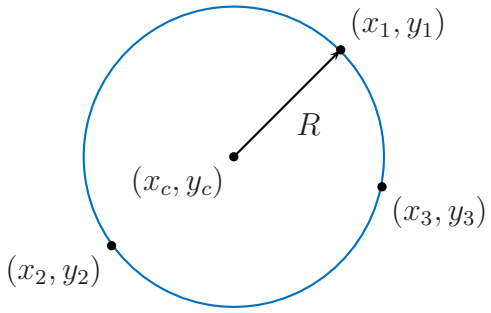

Figure 2: A circle fit through three points.

2 Fitting a circle through 3 points

Suppose you are given three points on a plane, (x_1, y_1), (x_2, y_2), and (x_3, y_3). As long as these points are distinct and not collinear, they define a circle, as shown in Fig. 2. Your task is to find the center (x_c, y_c) and radius R of this circle.

Finding such a circle requires formulating a set of equations to solve for (x_c, y_c) and R. You can easily eliminate R from these equations, and are then left with two linear equations in two unknowns to solve for x_c and y_c. You must solve these, and then create an algorithm that will implement your solution.

3 Specific Requirements

You must write a procedure named `circleFit`. This procedure must accept arguments of the *exact* types and in the *order* specified here:

```
void circleFit(double x1, double y1,
               double x2, double y2,
               double x3, double y3,
               double & xCenter, double & yCenter, double & R);
```

You must also write a `main` function which will prompt a human user for the six values $x_1, y_1, x_2, y_2, x_3, y_3$, *in that order*. `main` should then use `circleFit` to determine x_c, y_c and R, and write them to standard output (using `cout`) *in that order*.

You may find it useful to define some other routines as well, and you may want to use the functions `sqrt` and `fabs` declared by including the header file `cmath` (using `#include <cmath>`).

Your `circleFit` procedure does not need to deal with the case of coincident points, or points on a line. But it must correctly deal with *all* cases of three distinct points not on a line (i.e., if there is a circle through the three points, your code must find it).

4 The Stadium at Corinth

If you would like to check for the center and radius of the circle that defines the starting line at the Stadium in Corinth, here are the 21 measured points on the starting line. Try fitting a circle through several triples of these measured data. What problem do you see?

Table 1: The x, y coordinates of 21 points along the inner edge of the starting line at the stadium.

Data point	x-coordinate (meters)	y-coordinate (meters)
01	19.880	68.874
02	20.159	68.564
03	20.676	67.954
04	20.919	67.676
05	21.171	67.379
06	21.498	66.978
07	21.735	66.692
08	22.810	65.226
09	23.125	64.758
10	23.375	64.385
11	23.744	63.860
12	24.076	63.359
13	24.361	62.908
14	24.597	62.562
15	24.888	62.074
16	25.375	61.292
17	25.166	61.639
18	25.601	60.923
19	25.979	60.277
20	26.180	59.926
21	26.412	59.524

5 Submission Requirements

You code should be implemented in a single source file named `circleGames.cpp` and placed in your `hw2` directory by 5:00 PM on the due date.

6 Tools for checking your code

A reference implementation will be available as a Linux executable named `circleGamesRef` in the course directory `\afs\engin.umich.edu\course\w04\eng101\hagar\Tools`. A tester will also be available.