

Assignment 3

In this lab you will implement a two-dimensional geometric bounds checker in C++20.

This assignment is worth 10% of your final grade.

Late submissions will not be graded.

Setup

If you're off campus, you'll need to install the UCSC VPN: <https://its.ucsc.edu/vpn/campus-vpn.html>

SSH into one of the CSE teaching servers using your CruzID Blue credentials:

```
$ ssh <cruzid>@noggin.soe.ucsc.edu
or $ ssh <cruzid>@nogbad.soe.ucsc.edu
or $ ssh <cruzid>@thor.soe.ucsc.edu
or $ ssh <cruzid>@olaf.soe.ucsc.edu
```

Create a suitable place to work: **(only do this the first time you log in)**

```
$ mkdir -p ~/CSE111/Assignment3
$ cd ~/CSE111/Assignment3
```

Install the lab environment: **(only do this once)**

```
$ tar xvf /var/classes/CSE111/Assignment3.tar.gz
```

Make and test the skeleton system:

```
$ make grade
```

Confirm your code is compliant with the shape specification:

```
$ make compliance
```

Accessing the teaching servers' file systems from your personal computer

Your home directories on the teaching servers are UCSC AFS, i.e. they are the same as if you logged into a BSOE Workstation or the UCSC Unix Timeshare. You can edit files in character mode on the console using vi, but the recommended editor for CSE111 is Microsoft Visual Studio Code (<https://code.visualstudio.com/>) which many of you will already be using.

You will need to install this SFTP (Secure File Transfer Protocol) client into VS Code by downloading from here: <https://marketplace.visualstudio.com/items?suntobright.vscode-sftp>

If you have trouble following the instructions in that link, ask a TA for help.

Using the GIT revision control system

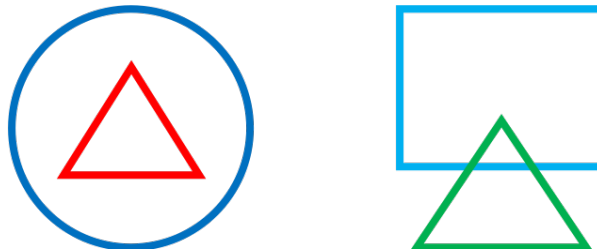
The CSE111 teaching servers have GIT installed and we highly recommend you use GIT to track the changes you make, though this is not required.

However, CSE111 is not a GIT training course; fortunately, Google is your friend, so find some beginner tutorials and study them if you have never used GIT. Also remember you have friends in the shape of your classmates, the TAs, and your instructor. If you want to know how to setup and use GIT, just ask someone.

Requirements

Consider two two-dimensional shapes, A and B, in the same plane. A is considered contained by B when the entire boundary of A is inside the boundary of B.

For example, in the following diagrams the red triangle is contained by the blue circle, but the green triangle is not contained by the cyan rectangle.



You are supplied with header files for three shapes: Circle, Polygon and Reuleaux Triangle. Your task is to write a bounds checker capable of determining if one instance of a shape is contained by another.

See the “Logic Sauce” section at the end of this document for more details.

Most importantly, **you must write your own tests to demonstrate your code works as expected.** A skeleton test for circles contained by circles is provided, **you should base your tests on that and by reading up on the Google Test framework:** <https://github.com/google/googletest>

What steps should you take to tackle this?

First, revisit the class recording of me implementing `Circle.ContainsBy(Circle &circle)` in class.

Order of implementation is yours to decide, but issues to consider include:

- The following C++ header files need completing:
 - `include/Circle.h`
 - `include/Polygon.h`
 - `include/Reuleaux.h`
- The following C++ source files need creating:
 - `src/Circle.cc`
 - `src/Polygon.cc`
 - `src/Reuleaux.cc`
- Additional tests need creating, for example:
 - `tests/CirclePolygonTest.cc`
 - `tests/CircleReaulexTest.cc`
 - etc.
- A sensible start might be as follows:
 - Complete `include/Circle.h`
 - Write `src/Circle.cc`
 - Uncomment the provided test in `tests/CircleCircleTest.cc`
 - Add more tests to `tests/CircleCircleTest.cc` to ensure your implementation of `Circle.ContainsBy(Circle &circle)` is working correctly.

Notes:

- Do NOT modify `include/Containable.h` or `include/Point.h` as they are not bundled in the submission archive; if your code depends on changes made it will fail to compile in the automated testing system and you will get no credit for this assignment.

How much code will you need to write?

A model solution that satisfies all requirements has approximately 180 lines of executable code not including test cases.

Running a subset of tests

If you just want to run test that have “Circle” in their name:

```
$ ./bounds --gtest_filter=*Circle*
```

Viewing the code coverage report

An HTML code coverage report is generated in the coverage folder. To transfer this to your workstation for viewing, do the following (note the “.” at the end):

```
$ scp -r <cruzid>@noggin.soe.ucsc.edu:~/CSE111/Assignment3/coverage/ .
```

You can then open `index.html` in the `coverage` folder locally.

Grading Scheme

The following aspects will be assessed by executing your code on a machine with an identical configuration to the CMPS111 teaching servers:

1. (100%) **Does it work?**

- a. Geometric calculations (50%)
- b. Every line of code in your implementation is covered by your tests (40%)
- c. Your implementation is free of compiler warnings and memory errors (10%)

Marks are deducted for any geometric calculations failing to produce the correct answer.

Note that in the grading system your code will be checked against known good tests. Some marks will be awarded for passing your own tests and gaining full code coverage. More marks will be awarded for passing the known good tests.

2. (-100%) **Did you give credit where credit is due?**

- a. Your submission is found to contain code segments copied from on-line resources and you failed to give clear and unambiguous credit to the original author(s) in your source code (-100%)
- b. Your submission is determined to be a copy of another CSE111 student's submission (-100%)
- c. Your submission is found to contain code segments copied from on-line resources that you did give a clear an unambiguous credit to in your source code, but the copied code constitutes too significant a percentage of your submission:
 - < 25% copied code No deduction
 - 25% to 50% copied code (-50%)
 - > 50% (-100%)

What to submit

In a command prompt:

```
$ cd ~/CSE111/Assignment3
$ make submit
```

This creates a gzipped tar archive `CSE111-Assignment3.tar.gz` in your home directory and checks it will execute successfully in the automated grading system.

If your submission is not compliant with the required shape interface, `make submit` will refuse to create the submission archive; you will need to fix your code and try again.

When you're happy that your submission is working as expected:

UPLOAD THIS FILE TO THE APPROPRIATE CANVAS ASSIGNMENT.

Secret Sauce

The most important things to remember when undertaking this assignment are the principles of advanced programming we're building up as we go through the class.

Keep it simple and take baby steps by starting with the simplest possible test you are supplied with - a small circle inside a larger circle. Get that working then move on to the next test, then the next, then the next. Pretty soon you'll have completed the whole mildly complicated assignment by taking a series of small steps.

You'll find all the necessary geometric calculation code on-line but may need to modify it slightly to fit your code, or possibly even convert from a different language. Having done this, do not forget to give credit to the original author(s) by including the source URL in your code as a comment.



Remember not to share code; feel free share ideas and useful sites you found in your investigations. You'll find collaboration is beneficial to everyone.

Have fun!

Scheduling Your Time

You only have a week to complete this assignment, so take care to work in a disciplined manner, concentrating on developing code that will gain you a good mark on the lab rather than functionality for which you have no tests.

Geometric Functions

As you go along, you'll develop a class of geometric functions that can be used by your (eventually) sixteen "type contained by type" functions like the `Circle.ContainedBy(Circle &circle)` function we developed in class. Eventually you will static methods to do things along the lines of:

- Calculate the distance between two points
- Determine if two line-segments intersect
- Find the centroid of a polygon
- Calculate the intersection points of a line-segment and a circle

And potentially others. Be rigorous in your factorization and take care to avoid duplicate code.

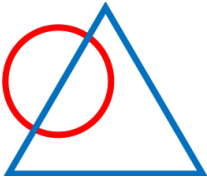
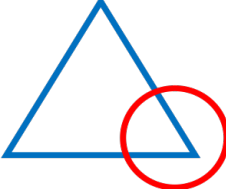
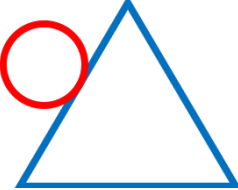
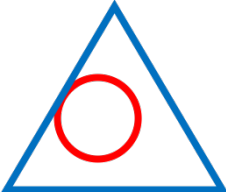
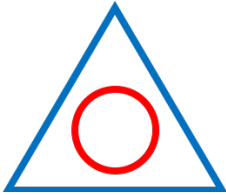
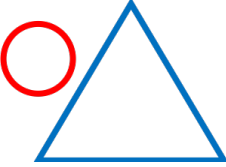
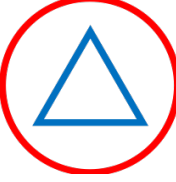
You may find this site useful for constructing more complex tests: <https://www.geogebra.org/geometry>

Logic Sauce

D if a circle is contained by another is easy; we simply calculate the distance between the centers of the two circles and evaluate whether this distance is less than or equal to the difference between the radius of the outer circle and the radius of the inner circle. A single calculation can take care of all possibilities.

For other shape combinations, circle inside a triangle, for example, you may find it easier to enumerate all the mechanisms by which a circle can be shown to NOT be contained by the triangle. If the circle can then not be shown to not be contained (a double negative) it must, by process of elimination, be contained.

Consider a circle that may or not be contained by a triangle and the existence of intersections between the circle's perimeter and the triangle's edges; the following possibilities exist:

	Intersections	Contained?	Potential Proof (there will be others)
	Two on the same edge	No	Show there are two intersections on any edge.
	One on each of two edges	No	Show two edges each have one intersection.
	One / Tangent	No	Show one edge has a single intersection and the center of the circle is further from the centroid of the triangle than the intersection is.
	One / Tangent	Yes	Show one edge has a single intersection and the center of the circle is closer to the centroid of the triangle than the intersection is. Or use the case below.
	None	Yes	Prove by showing all the negative cases do not hold. Also takes care of the previous case.
	None	No	Show no intersections and center of circle is further from centroid of triangle than the edge closest to the center of the circle.
	None	No	Show all vertices of the triangle are inside the circle. i.e., at a distance from the center of the circle less than or equal to the radius.