CS1112 Fall 2022 Project 1 due Wednesday 9/7 at 11pm

You must work either on your own or with one partner. If you work with a partner you must first register as a group in CMS and then submit your work as a group. *Adhere to the Code of Academic Integrity.* For a group, "you" below refers to "your group." You may discuss background issues and general strategies with others and seek help from the course staff, but the work that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is not OK for you to see or hear another student's code and it is certainly not OK to copy code from another person or from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

Objectives

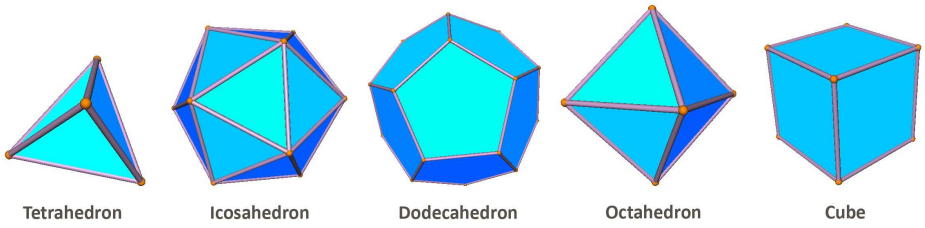
Completing this project will help you learn about MATLAB scripts, branching, boolean expressions, and some MATLAB built-in functions. You will also start to explore MATLAB graphics.

Ground Rule

Since this is an introductory computing course, and in order to give you practice on specific programming constructs, some MATLAB functions/features are forbidden in assignments: *only the built-in functions that have been discussed in class or are in this document or the assigned reading may be used.* In this project, *do not* use loops even if you know how to write one.

1 Platonic Solid (What???)

A solid is a *Platonic solid* if each face is identical in size and shape. There are five:



In the 16th century, the great Johannes Kepler tried to explain the orbits of the then six known planets by nesting the five Platonic solids with a sphere between pairs of platonic solids. For example, start with a sphere on the outside, then nest a tetrahedron inside, then nest a sphere inside, then nest a cube inside, and so forth, ending with a sphere inside.

Following in the footsteps of Kepler, we will determine some properties of the following nesting, from outside to inside, with spheres in between:

tetrahedron, cube, octahedron, dodecahedron, iscosahedron

By starting and ending with a sphere and nesting a sphere between each pair of platonic solids, we get six spheres. We assume that the outermost sphere is a unit sphere (i.e., radius one), the spheres have zero thickness, and the nesting is tight. Using the relevant formulas from Problem **P1.1.5** in *Insight Through Computing* (page 13), **write a script to calculate and print the radii and circumferences of the six spheres.** Read Problem **P1.1.5** in *Insight Through Computing* to learn how to use the relevant formulas but do not solve P1.1.5.

Through calls to function `fprintf` with appropriate format sequences, print the radii and circumferences of the six spheres from outermost to innermost through the 9th decimal place neatly in a "table format," i.e.,

the values should line up and be right-justified in columns. See, for example, scripts `Eg1_2` and `Eg1_1` in *Insight* for examples on lining up values in a column using `fprintf` and format sequences. Save your script file as `keplerSpheres.m`.

Here's an example of the *format* of the first few lines of expected screen output; the *values* shown below may not be right ...

```
Sphere          Radius          Circumference
--------------------------------------------------
  1           1.000000000        5.280000007
  2           0.555553333        3.903495102
```

2 Casting a Shadow

To create realistic looking CG (computer graphics) in games and movies, the lighting and the corresponding shadows, among other things, need to be properly computed and rendered. In this exercise, you will determine the extent of the shadow that will be cast by a box in a lit room. Consider the floor plan views—we consider the 2-d case only—below:

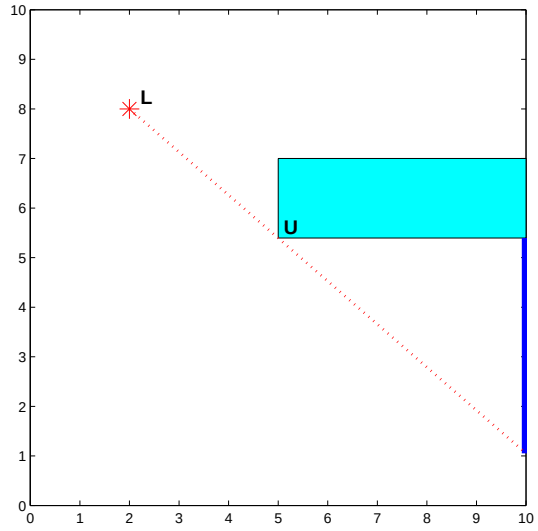


Figure 1. Shadow along right wall only

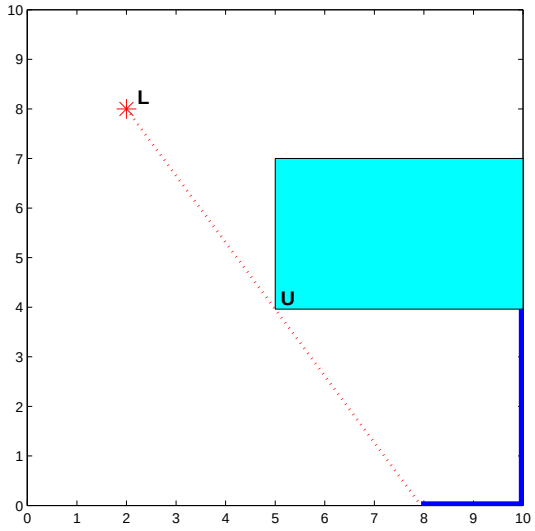


Figure 2. Shadow along right and bottom walls

With the box (rectangle) placed against the right wall and the point light source left of and above the top left corner of the box, the shadow along the right wall will extend from the bottom edge of the box to a point that is on either the right wall (Figure 1) or the bottom wall (Figure 2). The extent of the shadow depends on the position of the light source (point L) and the lower left corner of the rectangle (point U).

You will complete the script `castShadow` to position the light and the box in a room, determine the extent of the shadow, and produce the corresponding graphic similar to the figures above.

First, download the file `castShadow.m` and run it. A graphics window showing two points and two lines will pop up. The message near the top (the title area) says to click on the dotted line. After you click, a magenta cross marks the clicked point and its coordinates are shown in the title area.

Next, read the program to learn what it does. Don't worry about the early commands to set up the figure window, but here's how the `plot` statement works: `plot(x,y,'bo')` draws a marker at the point (`x`,`y`) with the format "blue circle"; `plot([x1 x2],[y1 y2],'k:')` draws a line from the point (`x1`,`y1`) to (`x2`,`y2`) with the format "black dotted line." Other formats are explained in the program comments. The statement `[xu,yu]=ginput(1)` accepts one mouse click by the user and stores the x- and y-coordinates of the click in the variables `xu` and `yu`, respectively. A statement `title('hello there')` would display the text 'hello

there' as the title of a figure. The `sprintf` statement works just like `fprintf` in formatting text, but instead of printing directly to the Command Window, `sprintf` allows the text to be saved under a variable name. Then this text variable can be used in other statements, such as the `title` statement as shown in the program.

The given program sets up a 10×10 "room" and demonstrates several commonly used graphics commands. **Here are the ways that you will need to modify the program.**:

1. Change the fixed location of the light source (point L) to randomly generated coordinates within the rectangular area that is left of and above the top left corner of the rectangle, point T. *Hint*: The statement `v = rand()` assigns to variable `v` a random number in the range of 0 to 1. So how do you get a random number within a different range? First, the statement `v = rand` gets you a real number in the range of 0 to 1. Next, scale (think multiply) and shift (think add) the value `v` as necessary to get the range that you need.

2. Let the y-coordinate of the user-clicked point indicate the bottom edge of the rectangle. The user is asked to click on the dotted line which extends vertically from point T, but since the click may not be precise enough, be sure to use only the y-coordinate from the user-clicked point and the x-coordinate of point T for the lower left corner of the rectangle. This is point U. The rectangle is then formed by points T and U and the right wall, which has the x-coordinate 10. Draw the four edges of the rectangle by plotting four solid lines in a color (or multiple colors) of your choice. (You can modify the code for the original cyan line to draw one of the edges of the rectangle.)

3. Determine the extent of the shadow cast by the box. There are different ways of solving this subproblem. For example, you can use the slope of a straight line. *Try to solve this geometric puzzle, but don't worry if you need a little help. In a couple days we will post more detailed hints on "the math," but we want to give you a chance to think through it on your own first.*

4. Plot a red dotted line from point L to the furthest extent of the shadow.

5. Mark the shadow along the wall(s) by drawing one or more thick blue lines. To produce a line of 3-point thickness (default thickness is .5 point), use the plot property `LineWidth`:
   ```
   plot([xL x2],[yL y2],'b-','LineWidth',3)
   ```

6. Change the message in the title area to give the coordinates of the light source instead of the clicked point. (You need to add another `title` statement.)

Note: your graphic will look a little different from the diagrams above. In particular, your rectangular box has four color edges but not a fill color, there are extra markers and an extra dotted line, and your graphic has a title message.

3 Quadratic Function

Complete exercise **P1.2.8** in Chapter 1 of *Insight*. Save the file as `quadExtrema.m`.

Submit your files `keplerSpheres.m`, `castShadow.m`, and `quadExtrema.m` in CMS (after registering your group).