CS1112 Fall 2015 Project 2 due Thursday 9/17 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 solidify your understanding of Chapters 2 and 3 in *Insight* (`for`-loops, `while`-loops, nested loops). You will also further explore MATLAB graphics.

Ground Rule

Do not use arrays or the `break` command in this project. In fact, do not use the `break` command in this course.

1 Another estimate of π

Read §2.1 of *Insight*. The "Talking Point" at the end of §2.1 (page 33) suggests that the approximation of π may be improved by including the number of cut tiles, B, in addition to the number of whole tiles, N. Write a script `tiles4pi` that counts every cut tile as a fraction f of a whole tile. Solicit as input f and the radius n. Display to the Command Window these values: n, f, N, B, the estimates of π using whole tiles only and using both whole and cut tiles, and the errors associated with those estimates.

Submit your script file `tiles4pi.m` on CMS.

2 Flipping an unfair coin

Ann and Bob flips an unfair coin in a game—tails shows up twice as often as heads. In each trial of the game, Ann and Bob each flips a coin twice—four flips in each trial. In each trial, Ann wins if she gets two tails while Bob wins if his two flips are different. If there is a tie, then neither player wins that trial. The game stops when one player has won five trials or after 30 trials, whichever happens first. Write a script `coinGame` to simulate this game. The output displayed in the Command Window should include the total number of trials played and the number of trials won by each player.

Think about the problem in steps! There are three main tasks:

1. Represent the unfair coin. Function `rand` gives a value in interval (0,1). How do you use this interval so that "tails happens twice as often as heads"? Use only function `rand` for random number generation in this problem—do not use other random number generators.
2. In *one* trial, determine who is the winner (if there is one).
3. Repeat the trials until a stopping criterion is reached. What values does the program need to keep track of?

Thought Question (i.e., you don't have to submit an answer): Each coin flip is unfair; is the game unfair? Do Ann and Bob have the same likelihood of winning?

Submit your script file `coinGame.m` on CMS.

3 "Best" approximation of π by tiling

We now return to Problem 1, the approximation of π by tiling a (quarter) circle. You will now determine what value of f is optimal—gives the smallest error—given a radius n. Use this approach: try the values $.01, .02, \ldots, .99$ for f and determine the associated approximation errors. The answer to our question is then the value of f that gives the smallest error.

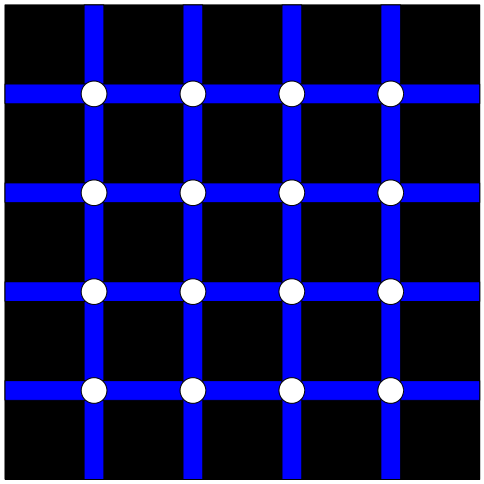
Start by saving your file from Problem 1 as `bestTiles4pi.m`. Then modify the code:

- Solicit for input only the radius n.
- For each candidate value of f, determine the approximation of π, the associated error, *and whether it is the best approximation so far*. If so, store the values of the best approximation so far, the best error so far, and the best f so far. In this way, by the time the program has tried all the candidate values, the "best so far values" are then the best overall. This algorithm requires that you *determine or assume* a value for the "best so far error" in the beginning. What value will you use? You can assume a reasonable value and write a comment to explain your choice. Check the MATLAB documentation for the keywords `realmin` and `realmax`. (You can use the *Help* menu option on the MATLAB desktop or just type, for example, `help realmin`, in the Command Window.)
- Make a plot of approximation error vs. f by plotting one point at a time as the program computes the approximation for each candidate f value. You will need these graphics commands:
  - `close all` to close all currently open figure windows
  - `figure` to start a figure window
  - `hold on` before you start working on the candidate f values (so that a subsequent plot command draws in the figure window without clearing previously drawn items)
  - `plot(x,y,'o')` for each point to be added to the graph, where `x` and `y` are the f value and the approximation error, respectively, and `o` is the marker format. You can choose any marker you like: `o` for a circle, `+` for a cross, `d` for a diamond, ..., etc. Don't forget to use single quotes to enclose the marker format character of your choice.
  - `xlabel`, `ylabel`, and `title` for labelling the x and y axes and for showing a title above the graph. For example, `xlabel('Fraction f')` will display the string `Fraction f` below the x-axis. The title should include the value of radius n. (See `triPartition.m` from Project 1 to see how to use the `title` and `sprintf` commands.)
  - `hold off` after the plot is complete (to return MATLAB to the default setting so that a subsequent plot command would clear the previously drawn items before drawing the current item)
- Display to the Command window the same values as specified in Problem 1 *but only for the overall best approximation*.

By running your script several times with different values for n, what can you say about the choice of n and f for approximating π?

Submit your script file `bestTiles4pi.m` on CMS.

4 The "Scintillating Grid"



Are your eyes playing tricks on you? Are some of those white disks in the diagram *flickering*? The diagram on the left is an optical illusion called the "Scintillating Grid." If you focus on a disk at a particular intersection, the neighboring disks appear to flicker. Cool! Now stop staring at it ...

Write a script `flickerGrid` that solicits n, the number of squares along each side, and draws the "Scintillating Grid." Assume that n is a positive integer value greater than 1. The example on the left has $n = 5$. Approximate the proportion that you see in the diagram and experiment with different colors to find your favorite scheme (that shows off the illusion).

Parameterize your code, i.e., identify the main properties (values) that define the diagram and name them as variables—parameters. Furthermore, choose *one* property to be the "root" and make the other properties dependent on it. For example, let's say that I see two important properties (there are more in this problem): gap width and square length. Instead of "hard coding" the variable values as `g=1; s=4;` I should make one variable dependent on the other, e.g., `g=1; s=4*g;` . Parameterization is an important concept in design and computational engineering. Choosing parameters wisely allows you to easily experiment with different values to "tune" your model and keeps your code easily maintainable.

Use the provided functions `DrawRect` and `DrawDisk`. Download the files from the course website and put them in the same folder as your script `flickerGrid`.

Graphics note: Use the figure window setup demonstrated in Lecture 6. Recall that `axis equal off` gives equal scaling in the horizontal and vertical axes and hides the axes labels.

Submit your file `flickerGrid.m` on CMS.