CS1112 Fall 2014 Project 6 Part A due Thursday 12/4 at 11pm

(Parts B and C will appear in separate documents. All parts have the same submission deadline.)

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, 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, please seek help from the course staff.

Objectives

Completing this project will solidify your understanding of structs and struct arrays (Part A), object-oriented programming (Part B), and recursion (Part C).

1 Rectangle structs

Note: This problem uses structure variables, not objects.

In computer graphics, the *ordering* of the items to be drawn needs to be correctly determined in order to produce a correct image representing the correct event. Consider a game example: the order *person-wall-orc* represents a far more favorable event than *wall-person-orc* ... We work with a simpler example here that orders a set of rectangles.

This problem follows the discussion in *Insight* §10.2 and makes use of the rectangle structure defined by the function `MakeRect`. Download the files `MakeRect.m` and `ShowRect.m` from the *Insight* page of the course website and use them in this problem. You will implement two functions to produce two figures, one showing randomly generated rectangles and the other showing the same rectangles drawn in the order large to small:

Figure 1

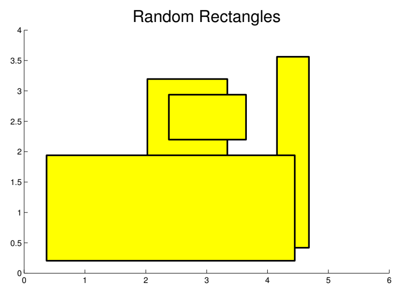
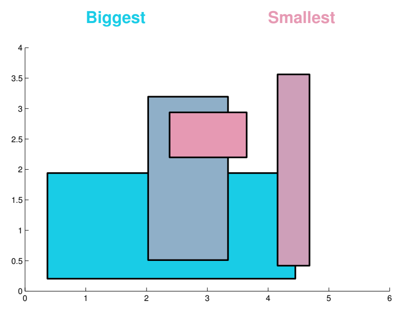


Figure 2



1.1 Random rectangles

Implement this function:

```
function Z = rectArray(n,xmax,ymax)
% Z is a 1-d array of n rectangle structs, each as defined by function
% MakeRect and each is randomly generated in the area bounded by (0,0),
% (xmax,0), (xmax,ymax), and (0,ymax).  The first generated rectangle is
% Z(1), the second generated rectangle is Z(2), and so forth.
```

1.2 Ordered rectangles

Implement this function:

```
function [colrZ, colrSorted] = rectangles(Z,xmax,ymax,colorS,colorL)
% Draw the rectangles in struct array Z:
% Figure 1 draws the rectangles in the order Z(1), Z(2), ..., Z(n), where n
% is the length of Z.  Draw all rectangles in one color of your choice.
% Figure 2 draws the rectangles starting from the largest (by area) to the
% smallest.  The colors of the rectangles are linearly interpolated with
% the largest area corresponding to colorL and the smallest area
% corresponding to colorS.
% colrZ is the n-by-3 matrix where colrZ(k,:) are the rgb values of Z(k).
% colrSorted is the n-by-3 matrix where colrSorted(i,:) are the rgb values
% of the ith smallest rectangle.
```

Figures 1 and 2 above are example graphical output from a call to functions `rectArray` and `rectangles`:

```
Z = rectArray(4,6,4);
[colrZ, colrSorted] = rectangles(Z, 6, 4, [.9 .6 .7], [.1 .8 .9]);
```

Sorting

Built-in function `sort` can be used to sort a numeric vector, i.e., a simple 1-d array of numbers. `sort` can be used in different ways:

```
[y,idx] = sort(x)  % y is x sorted in non-descending order and
                   % y(k) = x(idx(k))  for k= 1,2,...,length(x)

y = sort(x)  % y is x sorted in non-descending order
```

Linear interpolation

Recall that a linearly interpolated color c is a weighted average of two colors:

$$c = f \cdot \texttt{colorL} + (1 - f) \cdot \texttt{colorS}$$

where f is a fraction in [0,1] and is determined based on the *area* of the rectangle and not on the number of rectangles. For example, suppose there are three rectangles with areas 2, 4, and 10. Then $f = 0$ for the rectangle with area 2 and $f = .25$ for the rectangle with area 4.

Graphics

In the given function `ShowRect`, although the comments say that the color is one of the letters 'r', 'g', etc., you can instead use a vector of rgb values.

Label your figures as shown in the example above. For figure 2, write the word "Biggest" using `colorL` and the word "Smallest" using `colorS`. You may use built-in functions `title` and/or `text`. For example,

```
text(x,y, 'sometext', 'Color',[1 0 0], 'Fontsize',20)
```

will place `sometext` at position (x,y) in red using 20-point font. Function `title` works similarly but without specifying the position.

Start each figure with this code fragment:

```
figure
axis equal
axis([0 xmax 0 ymax])
hold on
```

After you have completed the graphics for one figure, use the command `hold off` to reset `Matlab` graphics to its default state.

Submit your files `rectArray.m` and `rectangles.m` on CMS.