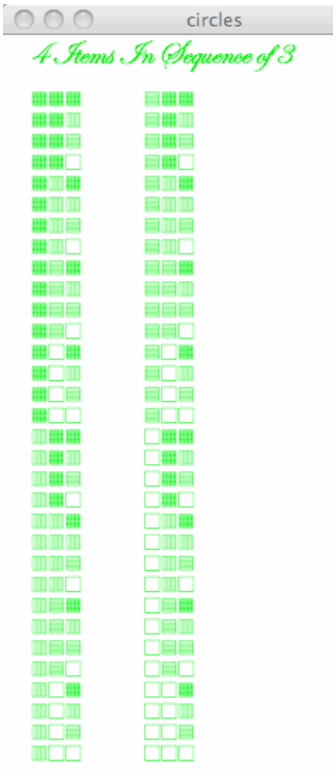
Homework 14: Enumerate Recursively



Goal: To enumerate four items in sequences of three using a recursive algorithm in Processing. There will be $n = 4^3 = 64$ total possibilities. You can enumerate squares or ellipses if you wish, but the easiest thing is to enumerate strings of characters. The solution at right prints the results in two columns.

Notice: This assignment does not explain in detail what to do; instead it expects you to use the hearts example from class as a guide (see below). So, the first step – really, don't bother reading further at this point – is to understand the hearts enumeration.

What To Do

Set up a standard processing program. (This one can be static.) Study the recursive enumeration presented in class, and notice how two String arrays are used. (String arrays were used in the last assignment, too, and are described in the Strings and Arrays Resource.) We need one array element for each of the 64 different configurations.

After setting up a standard Processing program, the following list of steps should do the job.

1) To have a heading, load a standard font of your choice.
2) To use UTF characters, like the blocks shown in this example, load a font of "special Unicode blocks". (Explained below.) It's like any other font and requires the same treatment – create, load, select – except instead of letters it has groups of similar UTF-8 symbols, such as the blocks shown in the example.
3) Because it is hard to type these special characters, grab them out of the file called `utf8Characters.txt` provided. Use them to initialize a String array like `binDigits[]`. (You can use other characters if you wish.)
4) Write a recursive function to enumerate these four items.
5) Return to the `draw( )` block and print out your results. A smart technique is to write a function (in Step 4) first that simply puts the first character into each element of the array, and prints it out.

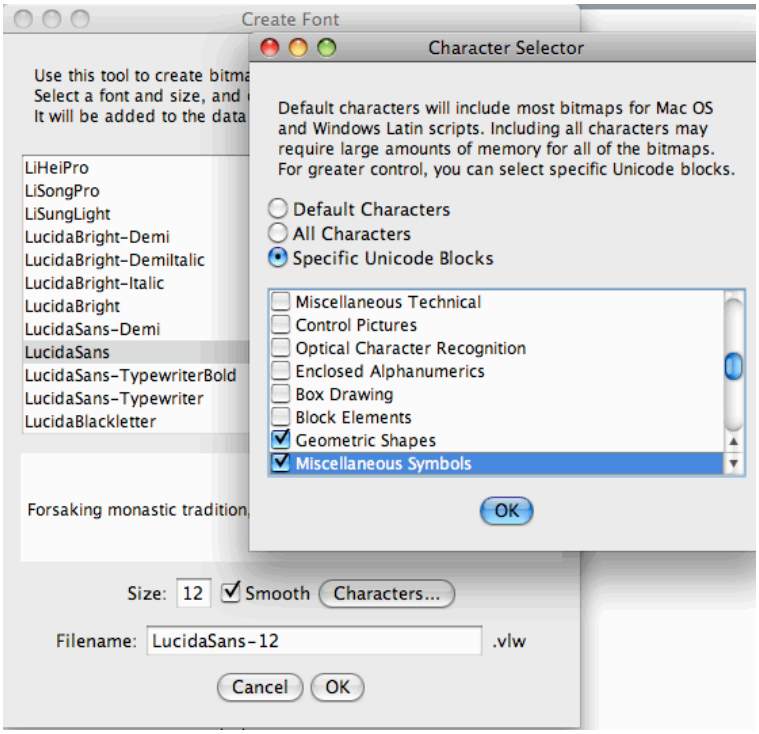
That's it!

Loading Special Characters

To get the fancy characters like the hearts or the blocks shown here, you need to load a special part of the UTF character space. To do this, navigate as follows in Processing

`Tools > Create Font … >` select a simple font; I chose `LucidaSan-12 > Character …`

Then scroll down past the fonts for many languages, until you get to the Miscellaneous Symbols and the Geometric Shapes. Click those and complete. See this screen shot:



Notice that you will have to switch between this font and your caption font when you want to change what you are printing out.

Wrap-Up

You have used a recursive Processing program (hearts) to guide your development of a recursive program to enumerate four items in sequences of 3. You have also noticed how it would be easy to modify this program to enumerate sequences of *k* items in sequences of *m*.

To Turn In

Submit your program to the class drop box, with its name changed to <yourname>.pde

Hearts Enumeration Code

```
PFont myFont, mySpecial;                    //Two fonts are needed
int n = 16;                                 //Two hearts in fours = 2^4
String[] seq = new String[n];              //The sequences are strings
String[] binDigits = {"♥", "♡"};            //The two letters to be displayed
```

```
void setup( ) {
  size(200, 550);
  myFont = loadFont("EdwardianScriptITC-24.vlw");//Get the caption font
  mySpecial = loadFont("LucidaSans-24.vlw");     //Get the UTF-8 block with hearts
  smooth();
  fill(255,0,0);                                 //Hearts are red
}

void draw( ) {
  background(255);
  smooth();
  textFont(myFont);                              //Prepare for the caption
  text("Putting My Heart", 20, 20);              //Caption, line 1
  text("Into Binary", 20, 45);                   //Caption, line 2
  textFont(mySpecial);                           //Prepare to display UTF-8
  for (int i=0; i<n; i++) {                      //Initialize with empties
    seq[i] = "";
  }
  addon(n, 0, "");                               //Build the String sequences
  for (int j=0; j< n; j++) {                     //And print them
    text(seq[j], 20, 80+j*30);
  }
}

void addon(int span, int base, String nextdigit) {
                                                 //span is the number of strings
                                                 //    that get the same character
                                                 //base is where those strings start
                                                 //    in the array
                                                 //nextdigit is the character that
                                                 //    is to be appended this time
  for (int i = 0; i < span; i++) {
    seq[i+base] = seq[i+base] + nextdigit;     //Set these items in the array
  }
  if (span > 1) {                                //Are we finished?
    addon(span/2, base, binDigits[0]);           //No, do one character in the first 1/2
    addon(span/2, base+span/2, binDigits[1]);    //And the other character in the second
  }
}
```