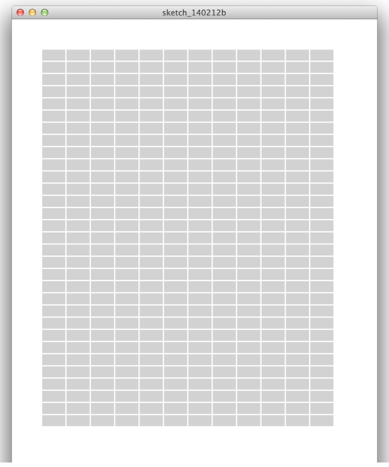
Assignment 14: Birthday Display

Goal

The goal is to practice with arrays and using them in a data-intensive info-graphic.

Warm Up



To get started write a Processing program that displays a 12 x 31 array of gray rectangles that correspond (roughly) to the days of the year, one column for each month. Try a canvas size that is 600x1000 and set `noStroke( )`.

This program will require a *nested loop*, which is a loop within a loop. We need two loops because one will move across a row of months, and the other will move down the column of days. We will draw 12 rectangles across, and then draw 31 such rows.

Begin by setting up a loop (loop variable `i`) to draw twelve 38 x 18 gray rectangles across the screen, starting 50 pixels in from the left and 50 pixels down from the top. Place them every 40 pixels.

Then put the following loop code AROUND the loop you just wrote; that is, the earlier loop will be INSIDE of this one:

```
for (int j = 0; j < 31; j++) {
                          ⟵      The "12" loop goes here
}
```

In the same way you adjusted the rectangle's horizontal positioning based on `i` above, moving across by 40 pixels per rectangle, you also need to adjust the rectangle's vertical positioning based on the `j` value, moving down by 20 pixels per row.

That's it, a nested loop – the "12" loop is nested inside of the "31" loop. Notice how it works. Save this file as `birthdaya.pde`.

Initializing Arrays

We know that an integer array `A` of three elements can be declared by

```
int[ ] A = new int[3];
```

but the array's elements don't yet contain any values. When you want to start out with values, replace the "`new int[`*<size>*`]`" with "{ *<initial items separated by commas>* }" as in

```
int[ ] A = {5, 4, 3};
```

This creates the array (the length being determined by the number of initial values) and fills-in the numbers in index order, i.e. `A[0]` has value `5`.

Find a file with this lab containing the data we will use. Declare three arrays at the top of the program, each initialized by the data given, named `month`, `day` and `rank`. Each array will have 366 elements. The data can be copy/pasted to be inside the curly braces.

Check the numbers. Because these are full tables, `month[0]` is 1 for January, `day[0]` is 1 for the 1st of January and `rank[0]` is its frequency – 364th, almost the least likely birthday! Obviously, Valentines days is found at 31+14-1 = 44 … we subtract one because arrays are indexed from 0. Save this file as `birthdayb.pde`.

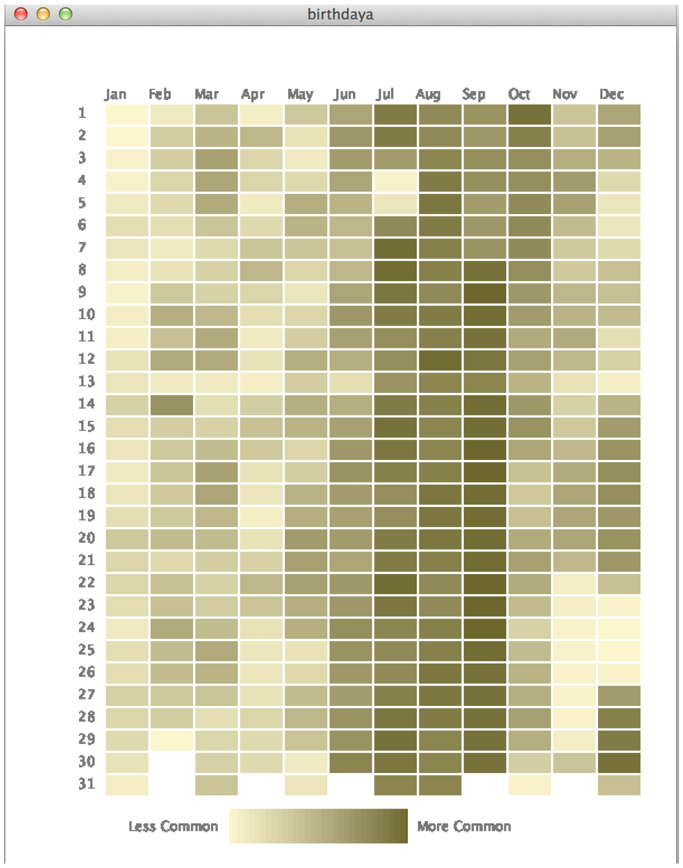
Color Range

We will use a color range to encode the frequency information so that the data is more understandable (see it at bottom of image). We create a key at the bottom by drawing a gradient as 160 vertical lines using the following code:

```
  for(int i=0; i<160; i++) {
    stroke(252-i, 247-i, 197-i);
    line(200+i, 700, 200+i, 730);
```

Notice how this works. The first line has `color(252,247,197)`, which is obviously quite light, and the last line has `color(252-159, 247-159, 197-159) = color(93, 88, 38)`, which is quite dark. So, our “spectrum” has 160 values.

Revise Warm-up Code To Use Data



The warm-up program above used two loops to draw the table, but we don’t need two now, because we have data arrays giving all of the values. Now, we just need one loop that ranges from 0 to 365 to cover each day of the year, and draw the correct rectangle when we find it in the list. Revise the outer `j` loop to cover that range.

Remove the inner loop, but keep the `rect( )` function.

Before the `rect( )` function use a `fill( )` function to set its color. The color will be chosen using the rank value, as follows. Because the rank covers a range 1 through 366, and because the color range is only 0 through 159, we need to find where in the spectrum the rank is. So, we multiply `160` times `rank[j]/366`. We see that this is right because if the rank were in the middle (`rank[j]=183`) then the fraction is 0.5 making the color increment 80, half way across the range. Write the code to go in the `fill( )`.

Finally, revise the rect( ) function. In drawing this box, you will not use the variables i and j as you did before. Instead, you will replace the month variable (i) by month[j], and the day variable (j) by day[j]. The rank is already being used for the color fill( ). Give it a try. Does it work?

Labeling

This info-graphic is nearly done. We need to label the columns and the rows. It is not necessary to import a font, although that is OK. Write the column headings with

```
text("Jan    Feb    Mar    …", 90, 65);
```

Label the rows with a loop on i containing the code

```
text(i+1, 65, 82 …);
```

In both cases there is something to fill in where the ellipsis (…) is. Finally, label the gradient bar with "Less Common" and "More Common" at its two ends.

Wrap Up: You have initially assigned values to arrays, and then used those arrays to display birthday data as an info-graphic. The technique used a color gradient to show rankings of the birthdays.

Turn In: Place your completed, documented code in the hw14 dropbox.