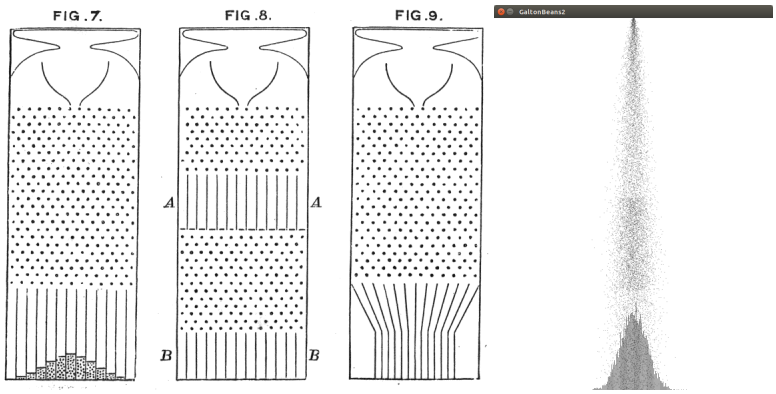
CMSC 157: Object-Oriented Programming Workshop

Assignment 3: Sir Galton's Bean Machine

Due by Class (1:30pm) September 19, 2016



The introductory chapter of the textbook describes the use of random numbers in simulation. Mostly, Shiffman discusses Uniformly (e.g., `random(0, 5)`) and Gaussian distributed (e.g. `randomGaussian()`) random numbers, but we are left asking: **how are these two types of random numbers related?** This assignment asks you to implement a simulation that shows how a normal coin flip (i.e., head or tails) can lead to the Normal distribution (i.e., Gaussian or Bell-curve). Sir Francis Galton imagined a machine[1] where marbles or beans start at the top and travel down through a series of pegs, going left or right at each peg. This simulation is related to one of the most important theorems in statistics known as the **Central Limit Theorem** and also the Pascal's Triangle and the binomial function.

Complete the `Bean` and `BeanMachine` classes. `Bean`'s `move` method should move the bean from the top to the bottom, and at each step, randomly chooses to move left or right. Processing's `constrain` function is useful for keeping the x-coordinate within `0` and `width`. The `step` method in `BeanMachine` should move each `Bean` and when a `Bean` reaches the bottom of the screen, it should add the bean to the count of beans at the `bottom`, and then `reset` the bean, so that it returns to the top middle part of the screen.

Learning Objectives

- Explore the Central Limit Theorem.
- Practice Implementing Classes.

Deliverable

Submitting your assignment:

1. Put a comment at the top of your programs with your name, date assignment description, and collaboration statement.
2. Bring a hardcopy of your program (i.e., the source code) to class.
3. Submit a zip file of your program via Moodle. The zip file should expand into a folder named `cmsc157-project3-lastname-firstname` with the Processing sketch inside of that folder.

[1] https://en.wikipedia.org/wiki/Bean_machine

```
1  /***
2  * Sir Francis Galton's Bean Machine
3  * Keith O'Hara <kohar@bard.edu>
4  * Bard CMSC 157 Project 3
5  * https://en.wikipedia.org/wiki/Bean_machine
6  */
7
8  BeanMachine bm;
9
10 void setup() {
11   size(600, 800);
12   smooth();
13   bm = new BeanMachine();
14 }
15
16 void draw() {
17   background(24);
18   bm.spawnBeans();
19   bm.step();
20   bm.draw();
21 }
22
23 class Bean {
24
25   // bean attributes here
26
27   Bean () {
28     reset();
29   }
30
31   void reset() {
32     // reset the bean to the middle of the very top of the screen
33   }
34
35   void move() {
36     // move the bean down, and to the left or right (equally likely)
37     // the bean shouldn't leave the screen
38   }
39
40   void display() {
41     // draw the bean
42   }
43 }
44
45
46
47
48
49
50
51
52
53
54
55
56
57
```

```
58
59 class BeanMachine {
60   int bottom[];
61   int numBeans = 0;
62   int maxBeans = 20000;
63   Bean[] beans;
64
65   BeanMachine() {
66     beans = new Bean[maxBeans];
67     bottom = new int[width];
68   }
69
70   void spawnBeans() {
71     for (int i = 0; i < 20; i++) {
72       if (numBeans < maxBeans) {
73         beans[numBeans++] = new Bean();
74       }
75     }
76   }
77
78   void step() {
79     // move each bean, and when appropriate add to the count on the
80     // bottom and reset the bean to the top
81   }
82
83   void draw() {
84     noStroke();
85     fill(255, 196);
86
87     for (int i = 0; i < numBeans; i++) {
88       beans[i].display();
89     }
90
91     for (int i = 0; i < bottom.length; i++) {
92       rect(i, height - bottom[i], 1, bottom[i]);
93     }
94   }
95 }
```