CS Major's Tea

- Monday 3/21 (Today)
- 4:30pm – 6pm
- Park 231

Animating with transformations

- basic object drawing MUST be centered on (0,0)
- variables: `x`, `y`, `size`, `angle`, `spin` etc
- display()

```
pushMatrix();
translate(x, y);
rotate(angle);
scale(size);
// drawing …
popMatrix();
```

- step()/move(): updates `x` and `y`
- spin(): updates `angle`
- keyboard/mouse callbacks in your main program should only update corresponding boolean variables
  - `spin = true;`
- Use callback functions
  - `mousePressed()`
  - NOT variables!

Example

- squareGrid

Jer Thorp. Artist/Educator - NYU

225 "random" numbers chosen and tweeted by 225 people

19	42	42	87	81	99	33	98	61	47	24	66
69	23	67	67	57	71	5	79	57	46	93	54
43	32	18	42	77	37	37	6	93	55	55	77
15	88	42	55	77	42	93	3	17	26	64	65
23	21	9	7	23	17	14	42	45	27	97	83
89	4	4	26	6	39	97	72	35	6	66	
19	2	72	81	37	47	66	17	12	52	74	
54	61	43	19	57	17	77	47	26	72	64	
69	99	64	88	67	1	36	2	60	27	73	
4	43	97	67	42	37	27	1	75	15	17	
13	59	32	78	40	15	64	77	11	1	17	
37	13	7	26	57	25	12	69	8	84	23	
66	42	14	33	17	97	25	57	1	81	97	
8	18	78	12	95	37	84	86	41	56	73	
78	60	21	39	28	17	83	69	12	74	37	
67	19	19	88	96	69	29	74	53	33	72	
32	81	72	72	73	39	52	97	77	77	41	
76	17	69	83	67	64	25	35	42	4	76	
13	36	2	37	52	47	43	25	66	7	6	
87	94	16	28	20	79	23	21	55	66	87	

http://blog.blprnt.com/blog/blprnt/your-random-numbers-getting-started-with-processing-and-data-visualization

loadStrings

- `String[] loadStrings(String)` is a built in function
  - takes a `String` as a parameter, interprets it as a file name
  - reads in the named file (in the same folder)
  - line by line and stores each line in an array of `String`, in order
  - returns the filled out array of `String`
- `String[] lines = loadStrings("data.txt");`

Raw Data

19
69
43
15
23
89
19
54
69
4
…

file name: data.txt
Stored in the **same** folder

Read Data File

```
void setup() {
  String[] lines = loadStrings("data.txt");
}
```

lines[0] = "19"
lines[1] = "69"
…

Data Type Conversion

- Variables of one type can be converted to other types.
- Type conversion function names are the types to which data will be converted

```
// binary(…), boolean(…), byte(…),
// char(…), float(…), str(…)

int i = int("200");
int i2 = int(lines[0]);

float f = float("1.23");
float f2 = float(lines[1]);
```

Modeling data

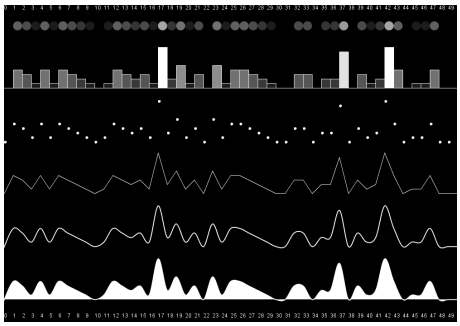
- A class to represent a data point
- Each object will represent an integer number 0-99
- Also contains a frequency counter

```
class Num{
  int num;   // value
  int count; // frequency

  Num(int num) {
    this.num = num;
    count = 0;
  }

  void inc(){
    count++;
  }
} // end class Num
```

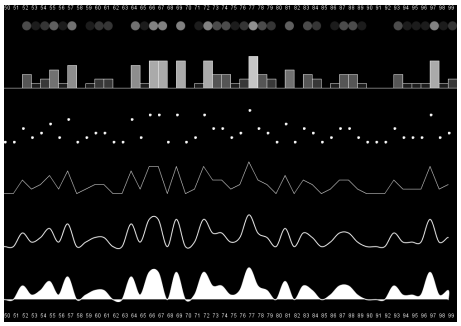
Read input and create data objects

```
Num[] readNumbers(String fileName){
  max = 0;
  String[] data = loadStrings(fileName);
  // create all 100 numbers, but with count set to 0
  Num[] nums = new Num[100];
  for (int i=0; i<nums.length; i++) {
    nums[i] = new Num(i);
  }
  // read input
  for (int i=0; i<data.length; i++) {
    // trim off white space (newline) and convert to int
    int n = int(trim(data[i]));
    // increment the frequency of this number
    nums[n].inc();
    if (nums[n].count > max) {
      max = nums[n].count;
    }
  }
  return nums;
}
```

Basic Plots 0-49



Basic Plots 50-99



Heat Map