<u>VIZUALIZATION TECHNOLOGIES</u>
<u>CALSS 8 NOVEMBER 6 2018</u>

ASSIGNMENT IN CLASS

Implement a program that enables you to place several vectors in the canvas, and shows two resulting average vectors:
– The first resulting vector (in red) is a traditional average, meaning that you add all the vectors currently placed and divide by the number of vectors.
– The second resulting vector (in blue) puts less weight on larger vectors, and more weight on smaller vectors. For this, while you are adding the vectors, each of them should be resized to 1000/currentMagnitude.

Each time you click on the canvas, a new vector is added from the center of the canvas to the current mouse coordinates. As you move the mouse, you should display the x and y coordinates in the reference system as illustrated in the movie “vectors.mp4”.

Notice that a vector is more than a line: it has a pointer. In order to draw a pointer, use the triangle() function. You will need to rotate that triangle to where the vector is pointing. For this use the heading() function in the p5.Vector class, and rotate the triangle by using the push(), translate(), rotate(), and pop(). Notice that because the angle that heading() returns has a different starting point than the one that the rotate() function uses, you will need to adjust this angle by adding HALF_PI to it.

When you press “d”, you remove the last vector from the list.

ASSIGNMENT 5

Explore a particle system similar to the one implemented in class 7. A certain number of particles should be initialized in random positions in the canvas (and not in the center as they were in class 7). The particles move not by a random step, but by a velocity which angle changes smoothly according with a noise function. You will initialize this velocity randomly, put you will update its angle according with a noise function. That way, you only have to update the position of the particle by adding the velocity to it: position.add(velocity). The documentation on the p5.Vector class is your friend: https://p5js.org/reference/#/p5.Vector

As you will see, you will use the p5.Vector.rotate() function to rotate the velocity:

e.g.

```
velocity.rotate(map(noise(frameCount/100), 0, 1, -PI, PI));
```

Additionally to each particle having its own velocity, they should also be attracted to nearby particles. For this, every particle looks at all other particles and computes an average vector that weights less farther particles. This is similar to the resulting blue vector done in class. To compute a vector that goes from particle A to particle B, you have to subtract vectors. A->B is B - A. e.g.

```
var ab = b.copy().sub(a); OR var ab = p5.Vector.sub(b, a);
```

When you compute this attraction force, you just have to add it to the velocity. Notice that these forces affect the particle behavior: if they are too small, nothing will happen, even they are too large, you particle will be projected very quickly to out the canvas and beyond. There is a sweet spot that has to be determined empirically, by trial and error, by adjust the resulting force vector with multiplications or divisions.

Notice that there is no right solution to this assignment. The objective is that you creatively explore the particles' behaviors as affected by other particles.