//////////////////////

DESIGNING SYSTEMS

//

BASES : L'ALÉATOIRE

//////////////////////

Sommaire

- La fonction **rand();**
- La Fonction **noise();**

Concepts / Mots Clés

aléatoire, randomness, chance,

La fonction rand()

Cette fonction calcule des valeurs 'pseudo-aléatoires'. Elle peut accepter 2 arguments. Si on ne lui attribue qu'une valeur, elle renvoie des valeurs entre 0 et la valeur.

```
random(100);
```

Dans l'exemple ci-dessus, la fonction random renvoie des valeurs entre 0 et 99.9999...*

```
random(5,80);
```

En occurrence, ici, elle renvoie des valeurs entre 5 et 79.9999...

*La fonction rand() renvoie une valeur du type flottante (0.003 par exemple). Si on veut calculer des nombres entiers, on a besoin de convertir en type int. Cette opération s'appelle 'casting' en anglais. Voir l'exemple Intro_Random_05 pour mieux comprendre ce procédé. Il est important car nous avons souvent besoin de convertir un type de donnée en un autre.

La fonction noise() [Voir les exemples dans le dossier f_NOISE]

La fonction noise, basé sur le travail de Ken Perlin (link en anglais), permet de générer des valeurs beaucoup moins erratiques et plus organiques. Pour cette raison, elle est souvent utilisé pour simuler des formes naturelles - tissus complexes, nuages... ou des mouvements fluides - l'eau / gaz / effets de vents ... Graphiquement, noise peut produire des effets très intéressants.

Elle ne fonctionne pas de la même manière que random. Premier point important à noter, c'est qu'elle renvoie des valeurs entre 0 et 1 seulement. Deuxième point important. Elle renvoie toujours la même valeur tant qu'on ne change pas l'argument entre parenthèse. Voir le sketch *Noise_01*

```
float n = noise(1);  // Ici nous déclarons une variable qui contient une valeur générée par la fonction noise. Elle sera toujours la même valeur.

print(n); // Regardez dans la console en bas du fenêtre Processing et vous verrez une seule valeur. Cette valeur est le nombre généré par noise. La fonction println permet d'afficher dans la zone du console en bas des informations sur une valeur. Elle est particulièrement pratique pour débugger lorsqu'on programme car nous pouvons voir exactement les valeurs de n'importe variable à chaque moment dans la lecture de notre sketch.

ellipse(n*width,200,20,20); // Ensuite pour positionner notre ellipse nous attribuons cette valeur 'n' et nous multiplions par un facteur égal à la largeur de notre espace de travail. Pourquoi ? Tout simplement pour positionner notre ellipse dans l'espace et par rapport à la la largeur. Essayez d'autres valeurs pour la fonction noise, par exemple noise(500). Elle renvoie toujours un nombre entre 0 et 1.
```

Dans le sketch *Noise_02* nous allons faire bouger notre ellipse pour dessiner une ligne avec la fonction noise. Alors, nous savons que noise renvoie toujours la même valeur donc pour faire varier cette valeur nous créons une variable, **noiseY**, que nous augmentons petit à petit, **noiseY+=0.001;**

Dans ce sketch, vous remarquez une autre fonction qui s'appelle **noiseDetail()**. Cette fonction détermine le comportement de noise. Plus la valeur entre parenthèse est élevé, plus les changements entre chaque valeur générée par noise sera important. Je vous conseille de jouer avec cette valeur et aussi avec la valeur de mise-à-jour pour la variable **noiseY.** Vous aurez des effets très différents à chaque fois.

Dans le sketch *Noise_03* nous allons créer une graphique de lignes qui suivent toujours le même chemin mais en décalant sa position sur l'axe Y nous pouvons créer remplir tout l'espace de travail. Alors, vous constatez que nous avons

déclaré deux autre variables ; **NoiseScale & noiseAmm** qui vont gérer l'échelle et la quantité de noise respectivement. Notez aussi que dans cette exemple nous utilisons directement la variable X (position de notre point) pour passer dans notre fonction noise. Ces valeurs vont de 0 jusqu'au 300 (la largeur de notre espace de travail). Noise alors calcule ses valeurs par rapport au position de notre point. Jouez avec les valeurs **noiseScale & noiseAmm** pour avoir des résultats différents.

Nous verrons des exemples plus avancée avec la fonction noise() plus tard. Il est important de bien comprendre son fonctionnement et la différence entre noise et random.