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

DESIGNING SYSTEMS

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

BASES : FONCTIONS

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

Sommaire

Les fonctions

Une fonction permet de créer des “actions” spécifiques dans un programme informatique.

Exemples de fonctions qui sont propres à Processing :

```
ellipse( 150, 200, 10, 10);

fill( 255, 0, 0 );

random( 200, 800);

pushMatrix();
```

Nous pouvons aussi créer nos propres fonctions en Processing pour faciliter des actions précises : dessiner/calculer/pivoter/récupérer... et en ce faisant rendre plus simple nos programmes.

« *En informatique, une fonction est une portion de code représentant un sous-programme, qui effectue une tâche ou un calcul relativement indépendant du reste du programme* » Source : http://fr.wikipedia.org/wiki/Fonction_informatique

La création se fait avec le mot clés ***void*** suivi par le nom de la fonction. Encore une fois, comme une variable, c’est nous qui désignons ce nom.

```
void choseBizarre () {

      //Les Instructions

      ellipse( 0, 0, 100, 50);

      ellipse( 0, 0, 50, 100);

}
```

Ensuite on fait appel à notre fonction dans **setup()** ou dans **draw()**.

```
void setup() {
	background(0);
	smooth():
	translate(100,100);
	choseBizarre();
}
```

L'intérêt des fonctions ? Réduire notre code et créer des 'actions' dédiées à une fonctionnalité spécifique. C'est ce que nous appelons dans le langage de la programmation, "*Abstraction*".

Créer des fonctions est un moyen d'abstraction. Autrement dit, l'abstraction permet de regrouper un certain nombre d'instructions selon des caractéristiques communs. Maintenant, nous n'avons pas forcément besoin de savoir comment cette fonction a été écrite. Ce qui nous intéresse principalement, c'est son utilité - que fait cette fonctionne - et éventuellement les arguments. C'est aussi la force de l'abstraction.

Maintenant que nous avons créer notre choseBizarre() nous pouvons le dessiner autant de fois que nous voulons sans réécrire toutes les instructions. Voir les sketchs *choseBizarre_02* & *choseBizarre_03.*

Sketches Star, Star_02 & Star_03 montrent une autre forme crée grâce à des fonctions et avec d'autres interactions.

Dans les autres exemples, notez bien que nous définissons également des arguments pour notre fonction. Souvenez-vous, dans Processing il existe beaucoup de fonctions qui acceptent ce qu'on appelle des arguments. Par exemple, ellipse(x,y,w,h) prend 4 arguments qui permettent de positionner l'ellipse et modifier sa taille en largeur et en hauteur.

Lors de la définition de notre fonction, nous avons également la possibilité de définir des arguments. C'est le cas dans les autres exemples qui suivent. Ce qui est important à saisir, c'est la logique. On définit les arguments nous mêmes. Lorsqu'on fait appel à une fonction dans setup() ou dans draw(), il ne faut pas oublier d'attribuer des valeurs pour ces arguments.

Voici l'exemple choseBizarre_02, j'ai marqué en rouge les arguments.

```
void setup() {
...
// *** valeurs pour les arguments x & y ***
 choseBizarre(100, 100);

...
}

/// FUNCTIONS ///////////
// *** définition des arguments x & y ***
void choseBizarre(float _x, float _y) {
  fill(255);
  pushMatrix();
  translate(_x, _y);   // *** arguments x & y ***
  ellipse(0, 0, 100, 50);
  ellipse(0, 0, 50, 100);
  popMatrix();
}
```