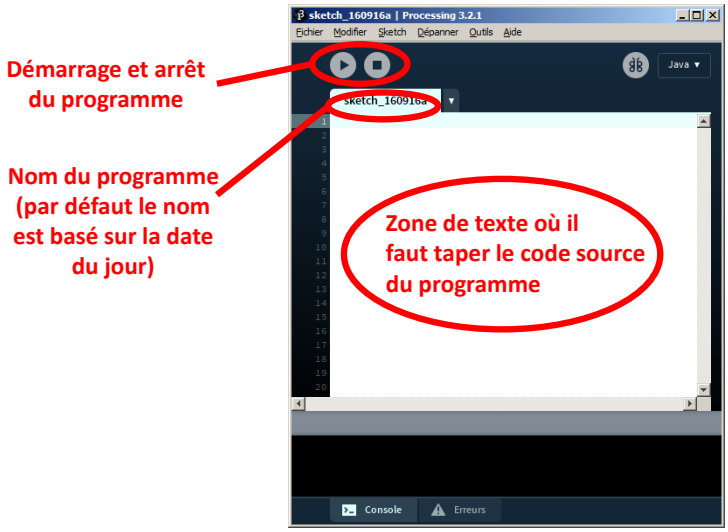
Séance Processing n°1 : Dessiner avec Processing

I./ Prise en main de l'environnement Processing :

L'*environnement de développement* Processing peut être téléchargé gratuitement à l'adresse suivante : http://processing.org/download/. Il suffit de sélectionner « No Donation » (ou un montant si on veut contribuer financièrement au projet) puis de cliquer sur « Download ». On choisit ensuite la version correspondant au système d'exploitation utilisé. On obtient alors une archive au format ZIP qu'il suffit de décompresser. Le logiciel ne s'installe pas, il suffit d'ouvrir le répertoire obtenu et de cliquer sur l'icône :

Après l'écran d'accueil, on arrive sur une fenêtre de ce type :

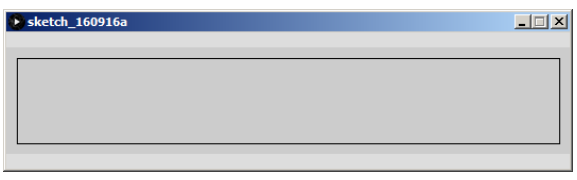


Dans la fenêtre blanche, recopier le texte suivant (bien penser à appuyer sur « Entrée » à la fin de chaque ligne) :

```
size(530, 100);
line(10,10,520,10);
line(520,10,520,90);
line(520,90,10,90);
line(10,90,10,10);
```

On remarque que les mots « size » et « line » sont colorés différemment du reste du texte. Ces deux mots sont en effet reconnus par Processing comme étant des *instructions*, c'est-à-dire des ordres exécutables par l'ordinateur. Pour faciliter la lecture du code source par un être humain, l'environnement de développement de Processing va utiliser différentes couleurs selon les fonctions des éléments du code source : c'est ce que l'on appelle la *coloration syntaxique*.

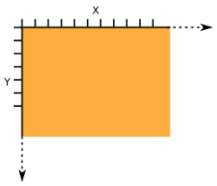
On appuie ensuite sur l'icône en forme de flèche vers la droite située au-dessus du code source : après quelques secondes, on obtient la fenêtre suivante :



Félicitations ! Vous venez de réaliser vote premier programme Processing !

II./ Quelques explications :

- Le code source est en fait une liste d'instructions que l'ordinateur va exécuter dans l'ordre, de la première jusqu'à la dernière.
- Chaque instruction est composé d'un *mot-clef* (dans notre exemple il s'agit de « size » et de « line ») suivi d'un ou de plusieurs *paramètres* entre parenthèses et séparés par des virgules.
- Chaque ligne doit être terminée par un point-virgule.
- Attention à bien respecter les minuscules et les majuscules : pour l'ordinateur ce sont deux choses totalement différentes.
- Le bouton en forme de flèche va lancer la *compilation* du code source, puis l'*exécution* du programme obtenu.
- L'instruction « size » définit la taille de la *fenêtre graphique*, c'est-à-dire de l'espace où on va pouvoir dessiner. Dans notre exemple cette fenêtre a une largeur de 530 pixels pour une hauteur de 100 pixels.
- L'instruction « line(x1, y1 x2, y2) » relie par une ligne droite les deux points de coordonnées (x1, y1) et (x2, y2).
- Les coordonnées du point le plus en haut et à gauche de la fenêtre graphique sont (0, 0). Les coordonnées x et y varient dans le sens indiqué par le schéma suivant :

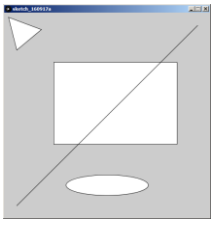


III./ Instructions graphiques de base :

Les *paramètres* (ou *arguments*) de toutes ces fonctions sont des nombres entiers. Par défaut (c'est-à-dire si on ne précise rien d'autre), le fond de la fenêtre graphique est gris clair. L'épaisseur des points, des lignes droites et des contours des figures géométriques est de 1 pixel. Les points, les lignes droites et les contours des figures géométriques sont noirs. L'intérieur des figures géométriques est de couleur blanche. On verra par la suite comment changer cela.

- `size(a, b)` : crée une fenêtre graphique de « a » pixels de large sur « b » pixels de haut.
- `line(x1, y1, x2, y2)` : trace une ligne droite reliant les points de coordonnées (x1, y1) et (x2, y2).
- `point(x, y)` : trace un point aux coordonnées (x, y).
- `triangle(x1, y1, x2, y2, x3, y3)` : trace un triangle dont on donne les coordonnées des trois sommets.
- `rect(x, y, a, b)` : trace un rectangle dont le coin supérieur haut a pour coordonnées (x, y), dont la largeur est « a » et la hauteur est « b ».
- `quad(x1, y1, x2, y2, x3, y3, x4, y4)` : trace un quadrilatère dont on donne les coordonnées des quatre sommets.
- `ellipse(x, y, d1, d2)` : trace une ellipse (un « ovale ») dont le centre est situé aux coordonnées (x, y) et dont le diamètre horizontal est de « d1 » pixels, et le diamètre vertical est de « d2 » pixels. Si d1 = d2, on obtient un cercle.

Exercice : à l'aide de ces fonctions, écrire un programme permettant d'obtenir le résultat suivant (la taille de la fenêtre est laissée à votre libre choix) :



Ne pas oublier d'enregistrer votre fichier sur votre répertoire de travail et sur votre clef USB.

IV./ Couleurs et mises en forme :

Blanc, noir, gris… Des traits fins… C'est un peu triste et monotone, tout ça. Voici quelques nouvelles instructions pour égayer tout ça !

- `background(nombre)` : définit la couleur du fond de la fenêtre graphique comme étant un niveau de gris égal à « nombre », sachant que 0 représente le noir, et 255 représente le blanc.
- `background(rouge, vert, bleu)` : rouge, vert et bleu sont trois nombres compris entre 0 et 255, qui définissent respectivement le niveau de rouge, de vert et de bleu dans la couleur du fond de la fenêtre graphique.
- `stroke(nombre)` : définit la couleur, en niveau de gris, des points, des lignes ou des contours des formes qui vont être tracés par la suite.
- `stroke(rouge, vert, bleu)` : définit la couleur, en rouge, vert et bleu, des points, des lignes ou des contours des formes qui vont être tracés par la suite.
- `fill(nombre)` : définit la couleur, en niveau de gris, du remplissage des formes qui vont être tracées par la suite.
- `fill(rouge, vert, bleu)` : définit la couleur, en rouge, vert et bleu, du remplissage des formes qui vont être tracées par la suite.
- `strokeWeight(pixels)` : définit l'épaisseur, en pixels, des points, des lignes ou des contours des formes qui vont être tracés par la suite.
- `noStroke()` : à partir de cette instruction, les lignes et les contours de formes ne sont pas tracés.
- `nofill()` : à partir de cette instruction, les formes géométriques n'ont pas de remplissage (elles se réduisent donc à leur contour).

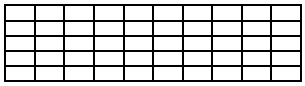
Remarques :

- Encore une fois, attention aux majuscules et aux minuscules (exemple des instructions « strokeWeight » et « noStroke »).
- Les instructions n'ayant aucun paramètre (comme « noStroke » ou « nofill ») doivent être suivies de deux parenthèses ().
- Si on utilise à la fois « noStroke » et « nofill », les figures n'auront ni remplissage, ni contour : elles ne seront tout bonnement pas tracées.

Astuce : pour obtenir les nombres « rouge », « vert », « bleu » correspondants à une couleur donnée, vous pouvez utiliser le site suivant : http://www.proftnj.com/RGB3.htm.

Exercices :

- Réaliser un dessin complexe et coloré en utilisant les instructions précédentes. Ne pas oublier d'enregistrer votre fichier sur votre répertoire de travail et sur votre clef USB.
- Ouvrir le fichier « Petite_maison.pde » situé dans le répertoire ressource (chemin donné par le professeur) et étudier le code source. Vous pourrez remarquer la présence de commentaires (ou remarques). Lorsque le code source devient long et complexe, on peut l'auto-documenter en plaçant de tels commentaires précédés de « // ». Modifier le code afin de rajouter des éléments au dessin ou pour modifier les éléments déjà présents. Ne pas oublier d'enregistrer votre fichier sur votre répertoire de travail et sur votre clef USB lorsque vous êtes satisfait du résultat.
- Faire un programme permettant de réaliser un quadrillage de 10 carreaux de large sur 5 carreaux de haut. Chaque carreau sera un petit rectangle de 20 pixels sur 10 pixels. Enregistrer le programme sous le nom « quadrillage.pde ».



Que faut-il faire si on vous demande de tracer un quadrillage de taille différente ? Nous verrons comment nous pourrons nous simplifier la tâche lors de la prochaine séance.

Listing du programme « Petite_maison.pde » :

```
// Petite maison

size(500,250);
background(0,255,255);  // Le fond de la fenêtre est cyan (bleu clair)
noStroke();             // Ne trace pas le contour des prochaines formes géométriques

// On dessine la pelouse
fill(0,255,0);          // couleur verte
rect(0,210,500,40);

// On dessine la maison
fill(128,128,128);      // couleur grise
rect(100,110,140,100);  // mur
rect(130,60,20,30);     // cheminée
fill(255,0,0);          // couleur rouge
triangle(170,50,240,110,100,110);    // toit de la maison
fill(142,91,0);         // couleur marron
rect(150,170,30,40);    // porte
fill(0,0,0);            // couleur noire
rect(170,190,2,4);      // poignée de la porte
stroke(0,0,0);          // on va tracer les contours (pour faire les bords de la fenêtre) en noir
fill(0,255,255);        // l'intérieur des fenêtres sera cyan
rect(110,120,40,40);    // tracé des deux fenêtres
rect(190,120,40,40);
line(130,120,130,160);
line(210,120,210,160);
line(110,140,150,140);
line(190,140,230,140);

// On dessine le sapin
noStroke();
fill(142,91,0);         // couleur marron
rect(350,170,30,40);    // dessin du tronc
fill(0,128,0);          // couleur vert foncé
triangle(365,40,340,80,390,80);     // tracé du feuillage
triangle(365,55,330,110,400,110);
triangle(365,70,320,140,410,140);
triangle(365,85,310,170,420,170);

// On dessine le soleil
fill(255,255,0);        // couleur jaune
ellipse(450,50,60,60);
```