Tutoriel - Programmer avec Processing (1)

TOUTES LES FICHES

par Yves Durasnel

PUBLIC	PARTICIPANTS	ANIMATEURS	NIVEAU	PRÉPARATION	ACTIVITÉ

Description

Découverte de Processing et de quelques programmes qui ne demandent pas la participation de l'utilisateur.("traitements par lots" ou "batch processing" en anglais)

Matériel

1 vidéo projecteur et 1 ordinateur maître
Accès internet non indispensable, processing doit alors être téléchargé et/ou installé au préalable
1 ordinateur par participant ou groupe de 2

Compétences travaillées

Concentration
Précision
Curiosité

Contenus utilisés

–

Pré-requis

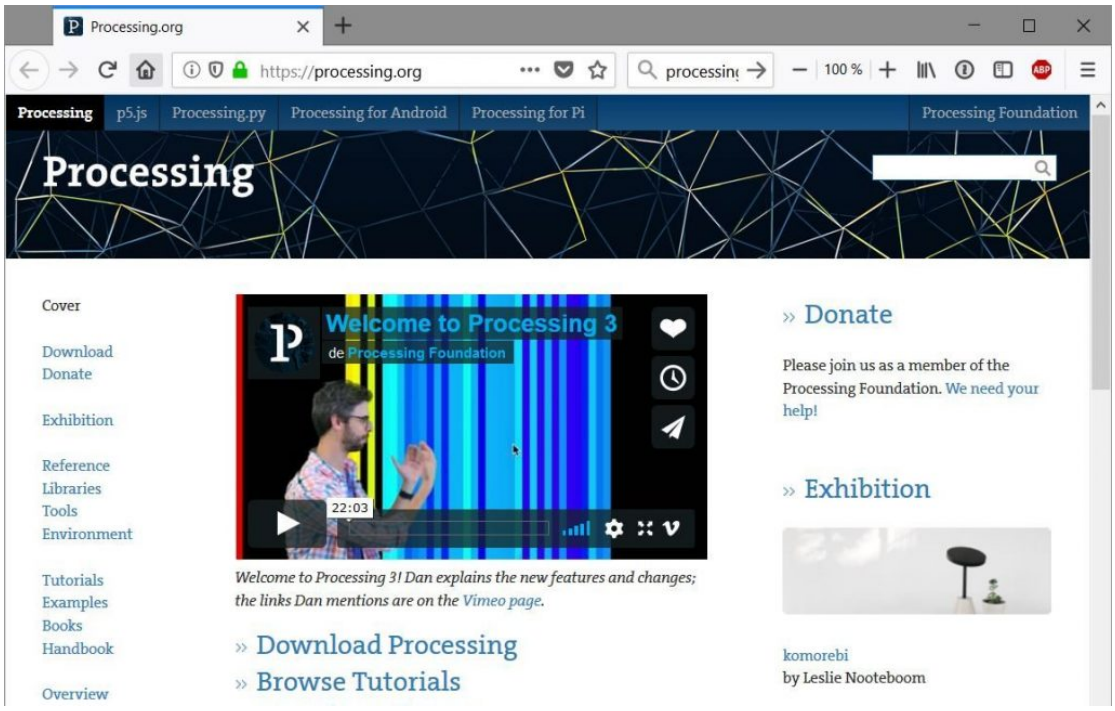
Avoir des bases d'anglais et savoir saisir du texte au clavier

WORKFLOW

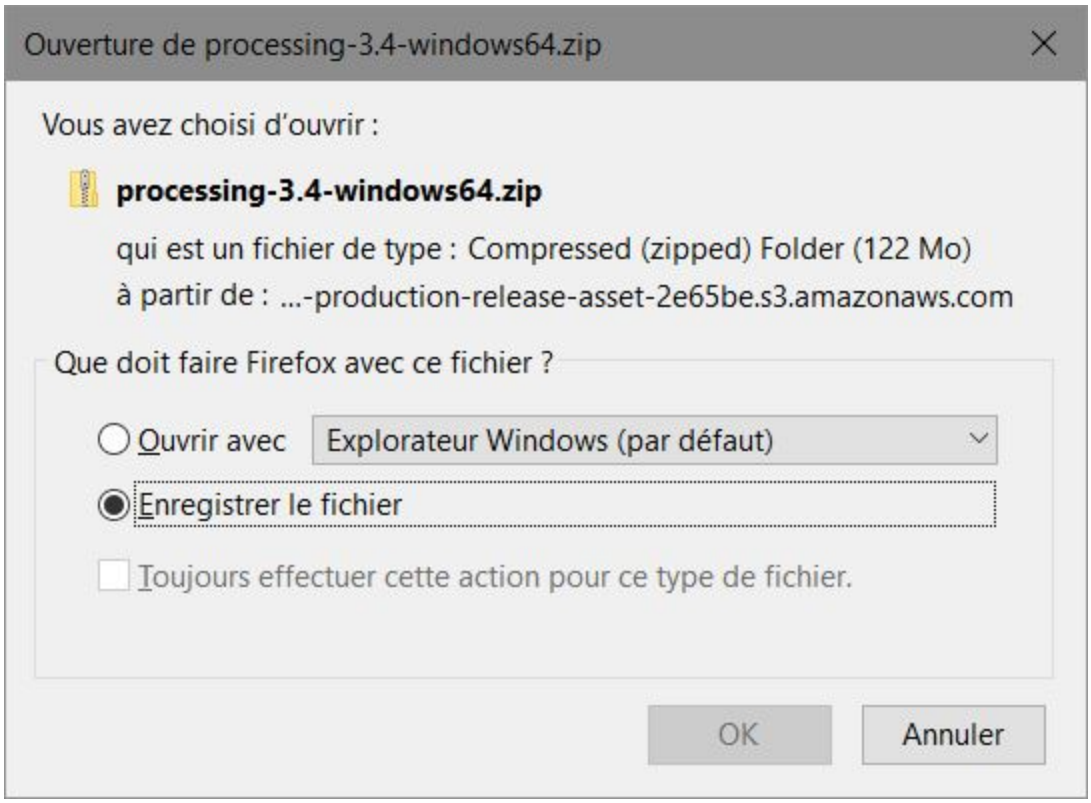
1 Installation

Processing est un environnement de programmation graphique développé par 2 artistes américains.

Le site de Processing est à l'adresse processing.org



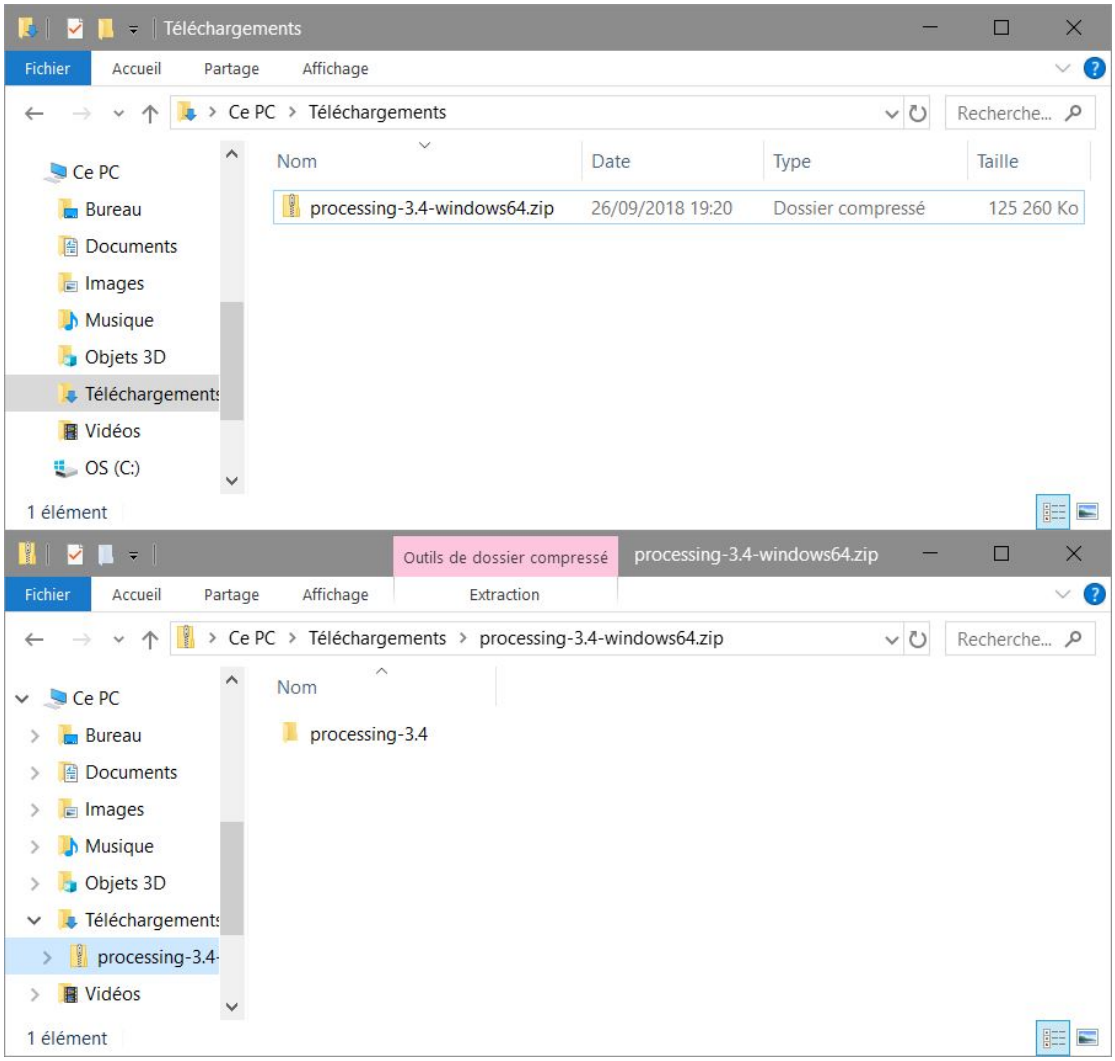
Le site est en anglais, mais le logiciel peut être configuré en français. Le téléchargement du logiciel se fait sur la page "Download".



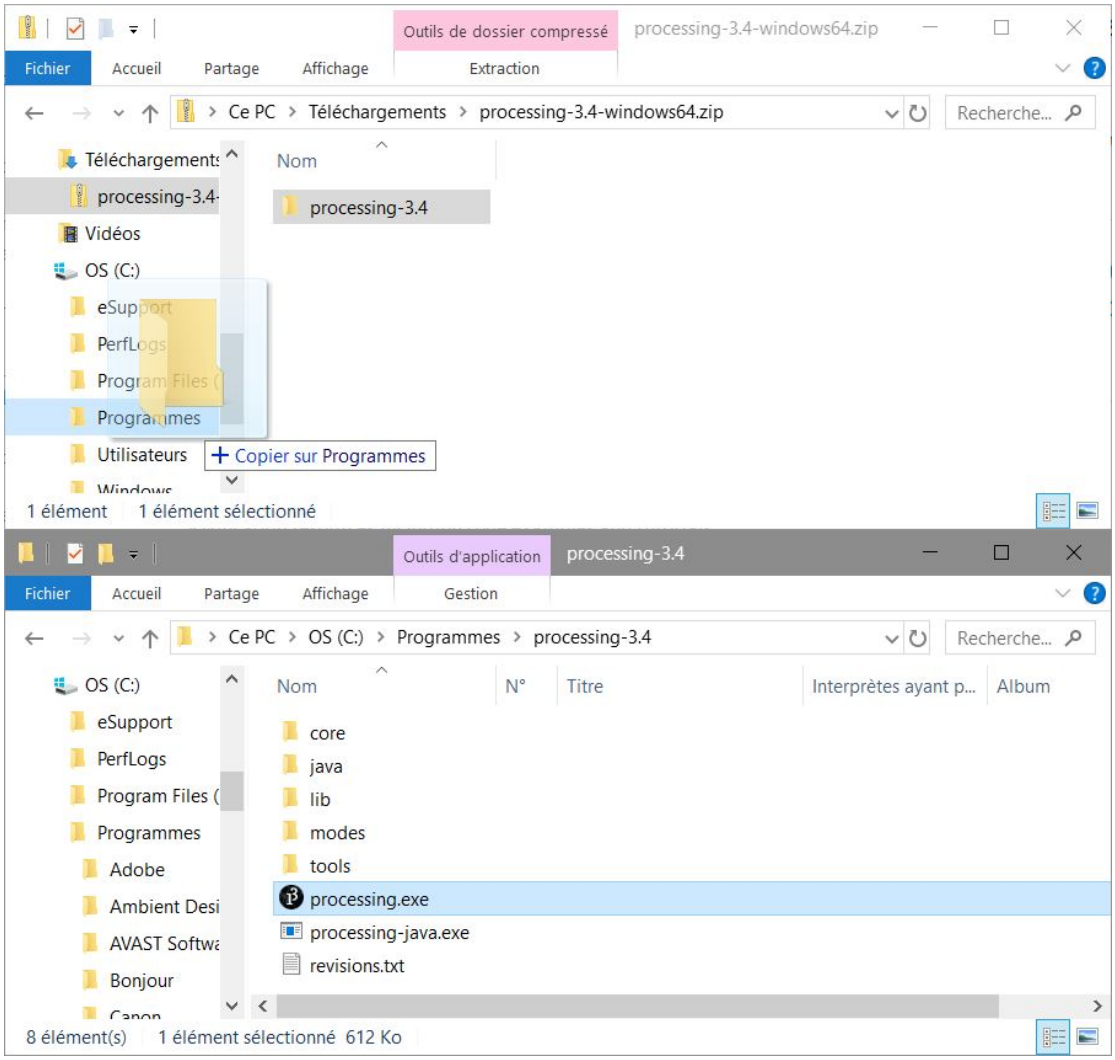
Sur Windows la version 64 bits est recommandée si le système correspond, ce qui est le cas sur les ordinateurs récents, sinon la version 32 bits s'installe sur tous les systèmes.

Le fichier téléchargé est une archive, le logiciel n'a pas de procédure d'installation automatisée.

Le répertoire compressé contenant les fichiers de l'application est simplement copié dans le répertoire "Programmes".

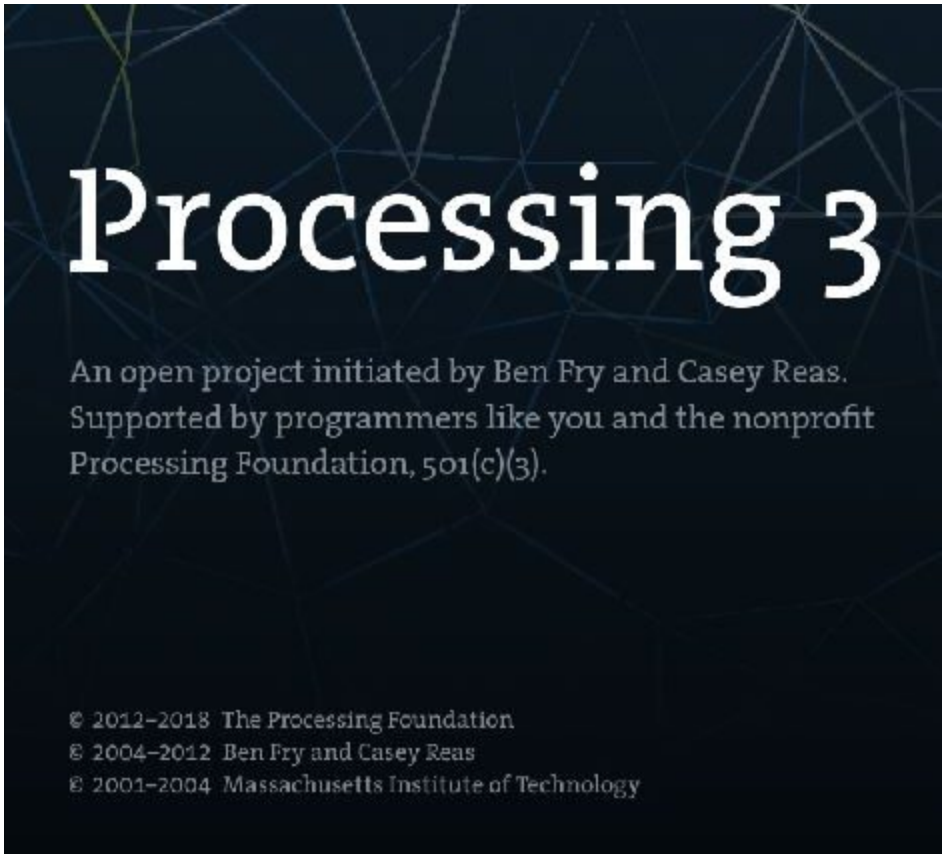


Ouvrez l'explorateur de fichiers. Dans le répertoire "Téléchargement" double-cliquez sur l'archive "processing-3.x-windows64.zip". Effectuez un "drag and drop" du répertoire "processing-3.x" dans le dossier "Programmes" du système (adaptez les indications à la version du logiciel en cours).

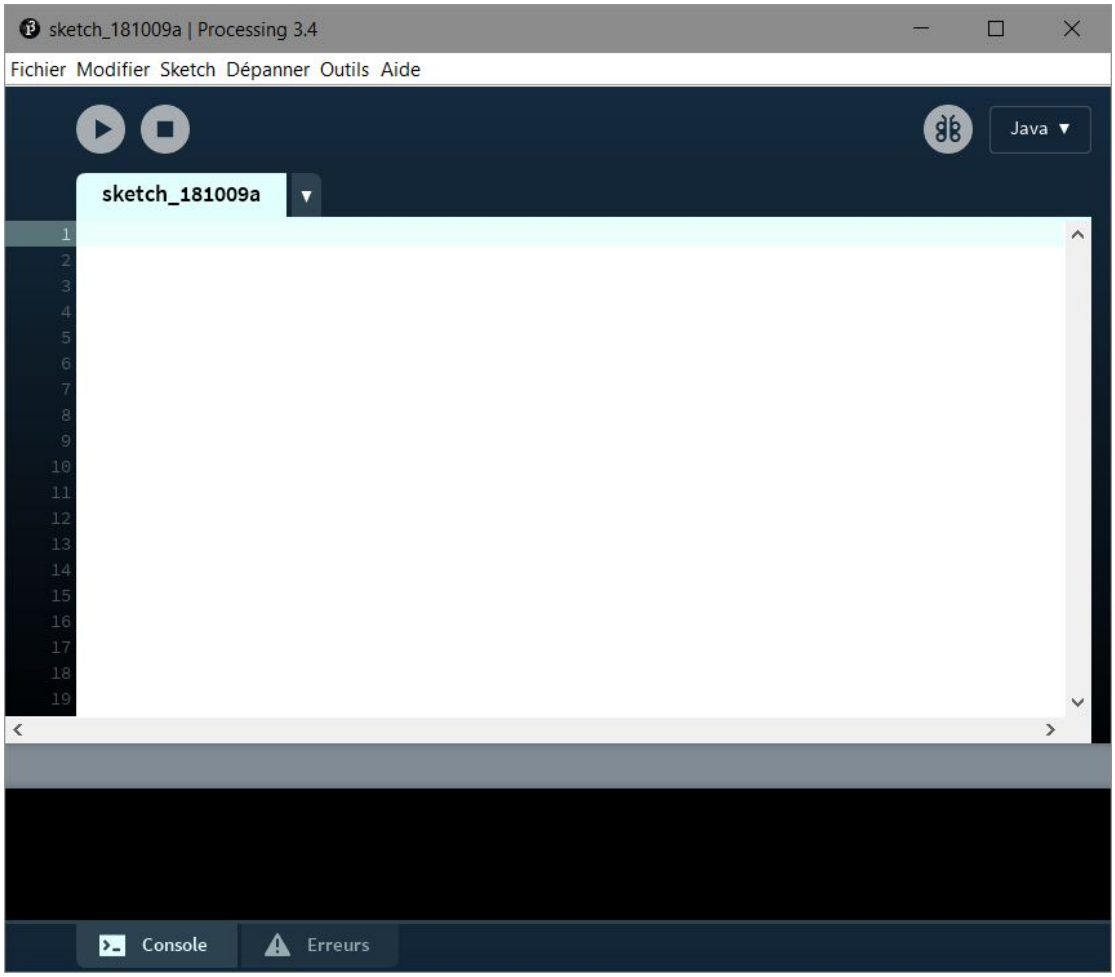


Enfin, double-cliquez sur l'exécutable "processing.exe" repéré par l'icone dans le répertoire "Programmes processing-3.x". Pour accéder plus facilement au programme vous pouvez l'épingler sur la barre des tâches ou la page d'accueil.

Premiers pas

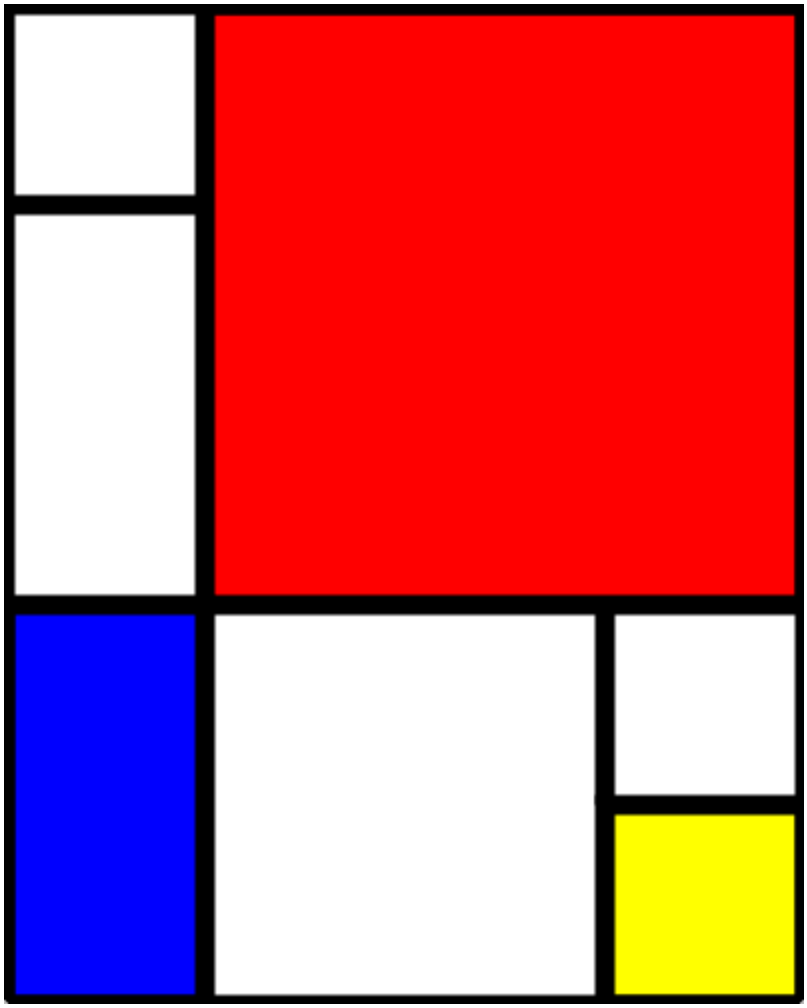


Au lancement un écran d'information s'affiche durant l'initialisation de l'environnement. La langue du système est reconnue et le logiciel est initialisé en français. Sinon la langue peut être modifiée dans le menu "Fichier Préférences".



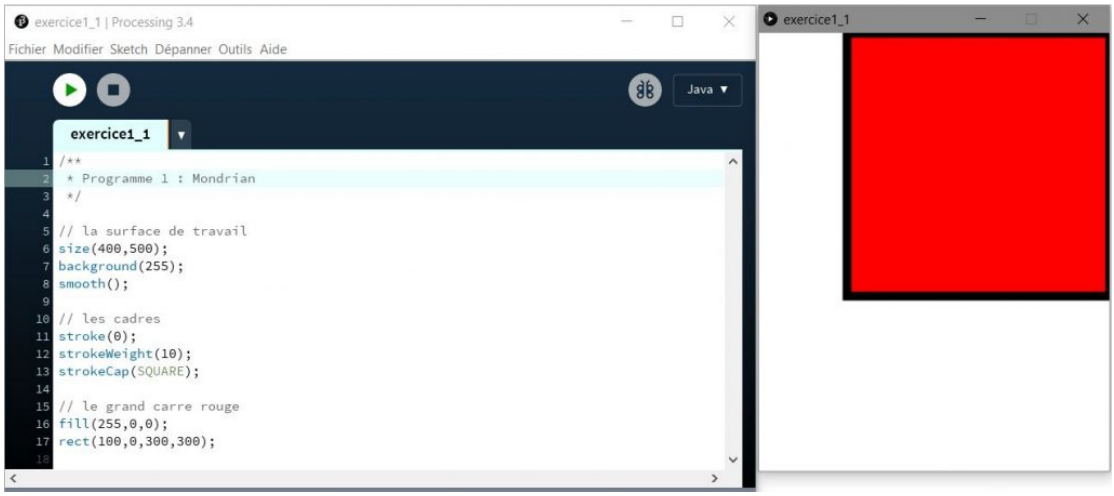
Au centre de la fenêtre est visible l'espace ou va s'écrire le code que nous allons développer, les lignes sont numérotées. En bas la fenêtre ou pourront s'afficher alternativement la console et les messages d'erreur utilisés pour vérifier le fonctionnement du programme lors de sa mise au point.

Premier programme : une image



Nous allons réaliser un hommage à Piet Mondrian, un peintre néerlandais reconnu comme un des pionniers de l'abstraction.

Le dessin, simplifié, est réalisé d'après le tableau « Composition en rouge, jaune, bleu et noir » réalisé en 1926.



A gauche, la fenêtre du code, à droite la fenêtre du résultat obtenu à l'exécution de ce code en appuyant sur le bouton démarrer 

Code utilisé :

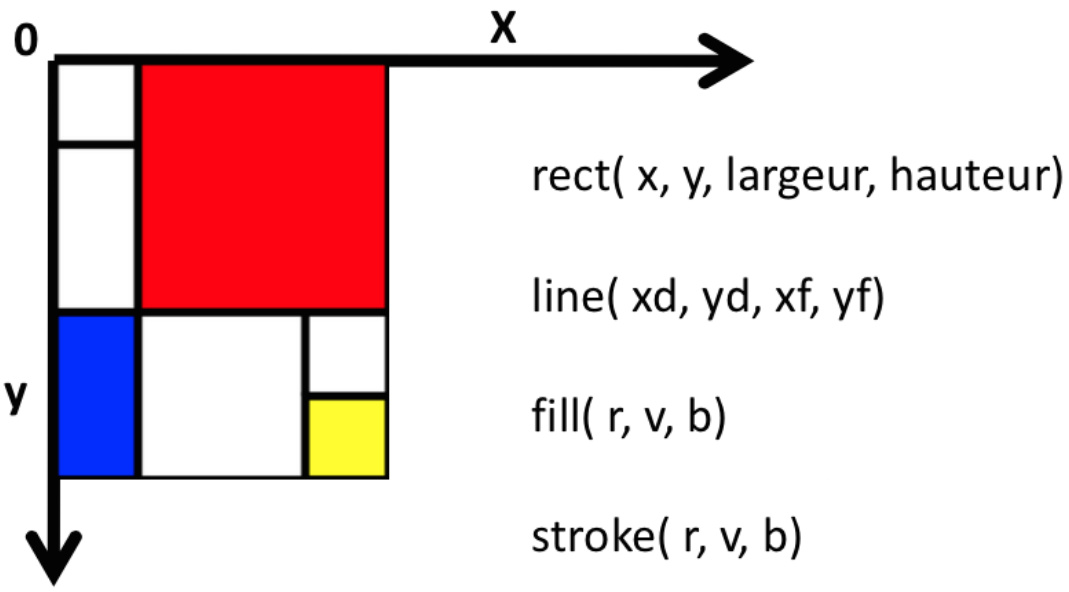
```
/**
* Programme 1 : Mondrian
*/
// la taille de la fenêtre de rendu
size(400,500);
background(255);
smooth();
// les parametres des cadres
stroke(0);
strokeWeight(10);
strokeCap(SQUARE);
// le dessin du grand carre rouge
fill(255,0,0);rect(100,0,300,300);
```

Analyse du code, remarquez :

- Les commentaires qui sont destinés à aider les développeurs, ils peuvent s'étaler sur plusieurs lignes, dans ce cas ils commencent par les signes /* et se terminent par les signes */ ou ne prendre qu'une ligne, dans ce cas la ligne commence par les signes //
- La coloration syntaxiques (les mots clés reconnu par l'application sont affichés en couleur)
- Les lignes de code sont toujours terminées par un point-virgule ;
- L'instruction size() est la première du code, elle indique la taille de la fenêtre d'exécution
- Les couleurs sont définies par 3 valeurs en notation RVB (consultez par exemple codes couleurs HTML pour en savoir plus)
- Il est indispensable de se référer à la documentation (menu "Aide Référence")

pour obtenir la liste des fonctions et les arguments utilisés.

Quelques primitives graphiques :



La fonction rect() permet de dessiner un rectangle défini par ses coordonnées (point en haut à gauche) et ses dimensions (largeur, hauteur).

La fonction line() permet de dessiner une ligne définie par ses coordonnées de départ et d'arrivée.

La fonction fill() permet de définir la couleur d'une forme.

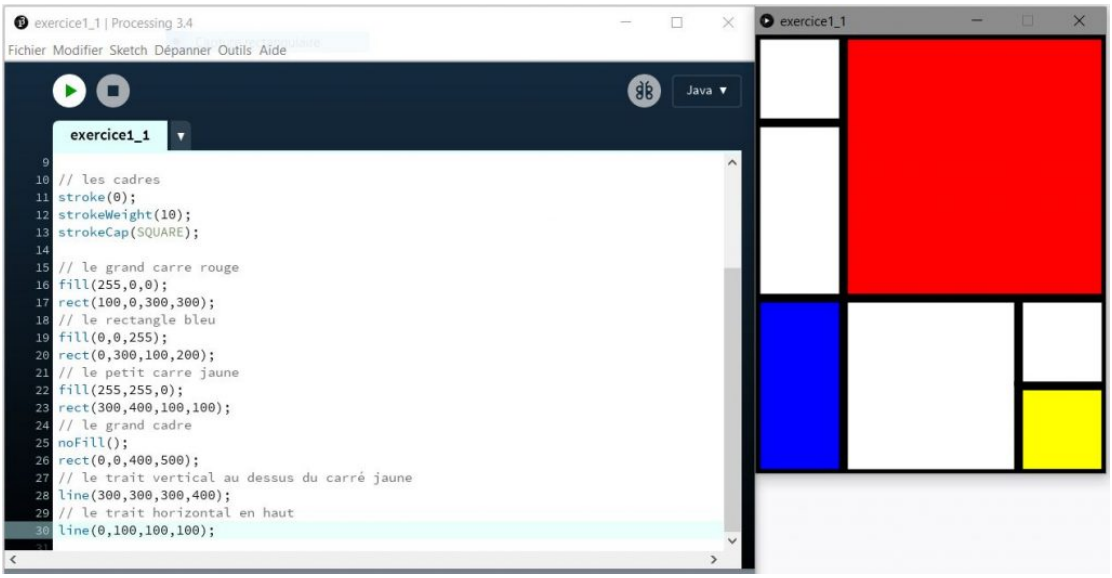
La fonction stroke() permet de définir la couleur et l'aspect du cadre autour d'une forme.

Le code complet :

```
/**
* Programme 1 : Mondrian
*/
// la surface de travail
size(400,500);
background(255);
smooth();
// les cadres
stroke(0);
strokeWeight(10);
strokeCap(SQUARE);
// le grand carre rouge
fill(255,0,0);
rect(100,0,300,300);
// le rectangle bleu
```

```
fill(0,0,255);
rect(0,300,100,200);
// le petit carre jaune
fill(255,255,0);
rect(300,400,100,100);
// le grand cadre
noFill();
rect(0,0,400,500);
// le trait vertical au dessus du carré jaune
line(300,300,300,400);
// le trait horizontal en hautline(0,100,100,100);
```

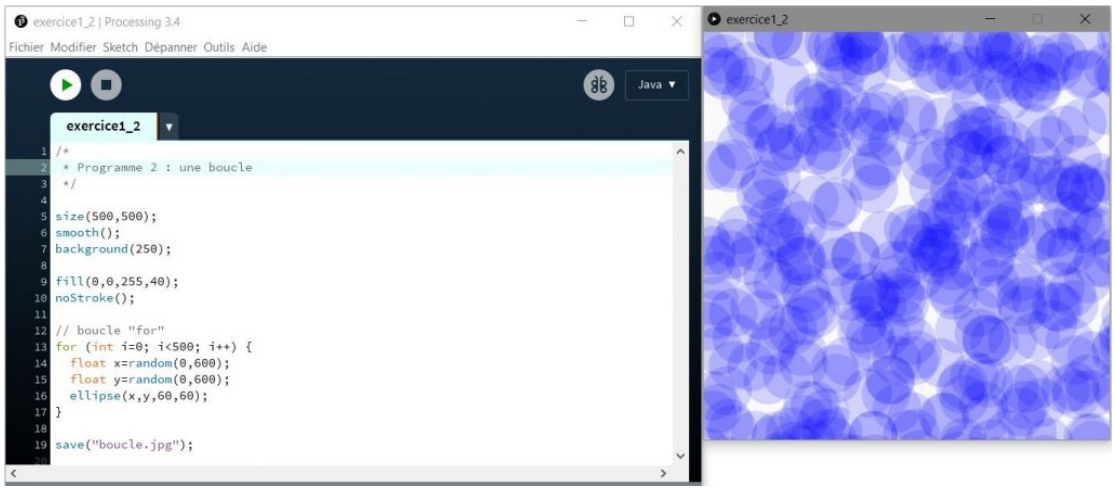
Résultat



Deuxième programme : une boucle

L'utilisation de l'ordinateur a pour but la réalisation de tâches répétitives. Réaliser une image statique comme précédemment a finalement peu d'intérêt. Maintenant nous allons réaliser un programme qui exécute 500 fois la même opération, dessiner des ronds bleus transparents à 40% à différents endroits de la fenêtre choisis de façon aléatoire.

Résultat



Code utilisé :

```
/*
* Programme 2 : une boucle
*/
size(500,500);
smooth();
background(250);
fill(0,0,255,40);
noStroke();
// boucle "for"
for (int i=0; i<500; i++) {
    float x=random(0,600);
    float y=random(0,600);
    ellipse(x,y,60,60);
}
save("boucle.jpg");
```

Analyse du code, remarquez :

- La boucle for peut s'expliciter ainsi

```
répéter en boucle (suivant un compteur de 0 à 500) {
    les instructions à exécuter;    }
```

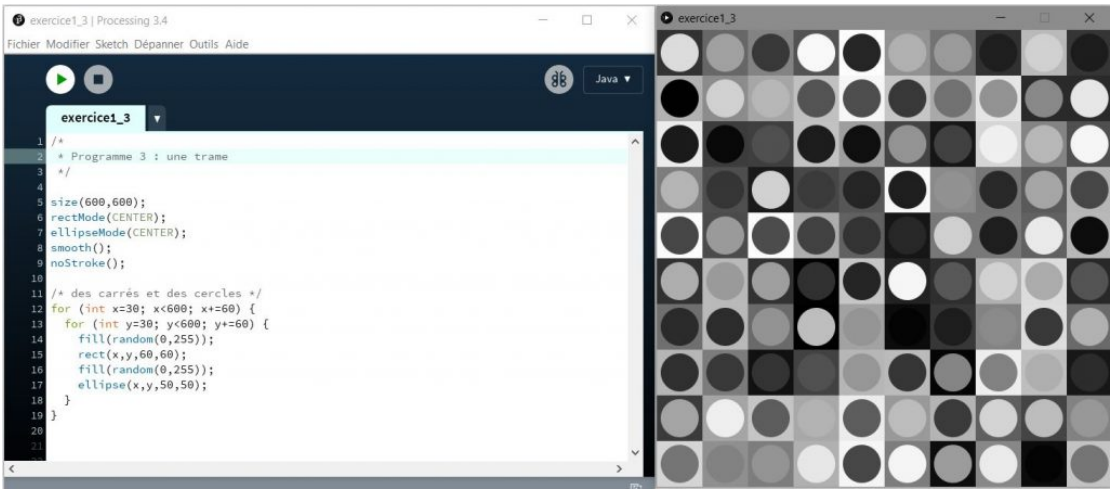
- La condition consiste à créer une variable i=0 et à l'incrémenter jusqu'à 500
- Les instructions à l'intérieur du bloc for sont incluses entre crochets {} et indentées (décalées par rapport à la marge pour rendre le code plus lisible)
- La déclaration de variables numériques (i, x et y) et la définition de leurs type int (entier) pour i et float (decimal) pour x et y
- L'instruction random() qui permet de générer une valeur aléatoire
- L'instruction save() qui permet de sauvegarder l'image en fin d'exécution

Troisième programme : une trame

En imbriquant deux boucles for nous allons créer un damier, chaque case contenant un cercle. La couleur des carrés et des cercles et fixée aléatoirement.

Dans le code proposé le damier est rempli colonne par colonne de gauche à droite, le sens de construction pourrait être inversé le damier étant rempli ligne par ligne de haut en bas.

Résultat



Code utilisé

```
/*
* Programme 3 : une trame
*/
size(600,600);
rectMode(CENTER);
ellipseMode(CENTER);
smooth();
noStroke();
/* carrés et cercles */
for (int x=30; x<600; x+=60) {
    for (int y=30; y<600; y+=60) {
        fill(random(0,255));
        rect(x,y,60,60);
    }
    fill(random(0,255));
    ellipse(x,y,50,50);}
```

Analyse du code, remarquez :

- L’instruction fill() ne reçoit qu’une valeur au lieu de trois, dans ce cas la valeur est répétée trois fois ce qui permet de définir un niveau de gris au lieu d’une

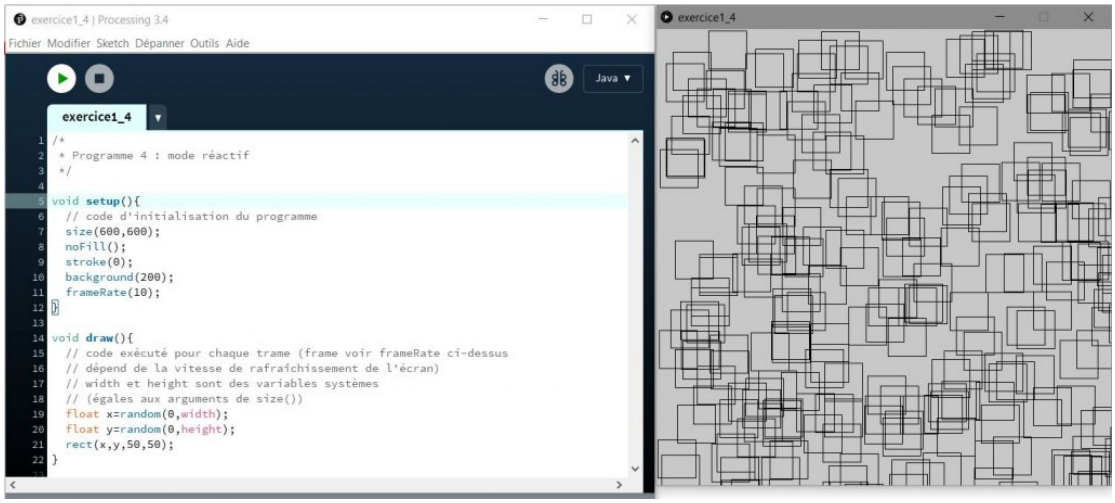
couleur.

- Les instructions rectMode() et ellipseMode() permettent de définir le mode de tracé des formes (à partir de leur centre dans ce cas).
- Un rectangle rect() dont les deux côtés sont égaux donne un carré, de même une ellipse ellipse() dont les deux rayons sont égaux donne un cercle.

Quatrième programme : mode réactif

En mode réactif l'affichage ne se fait pas une fois le code exécuté, comme dans les exercices précédents, mais est enrichi à chaque rafraichissement de l'écran de l'ordinateur.

Résultat



Code utilisé

```
/*
* Programme 4 : mode réactif
*/
void setup(){
// code d'initialisation du programme, exécuté une seule fois
size(600,600);
noFill();
stroke(0);
background(200);
frameRate(10);
}
void draw(){
// code exécuté pour chaque trame (ou frame voir frameRate ci-dessus
// dépend de la vitesse de rafraîchissement de l'écran)
// width et height sont des variables systèmes
// (égales aux arguments de size())
float x=random(0,width);
float y=random(0,height);
```

```
rect(x,y,50,50);}
```

Analyse du code, remarquez :

- Deux fonctions prédéfinies doivent être renseignées :

```
Void setup() {
    // code exécuté une fois à l'initialisation du programme
}
Void draw() {
    // code exécuté autant de fois que défini chaque seconde
}
```

- L'instruction frameRate() indique le nombre de rafraîchissements par seconde, il ne peut pas être supérieur à la capacité de la carte graphique et de l'écran.
- Les variables système width et height sont initialisée au démarrage du programme respectivement à la première et deuxième valeur de size().
- Pour stopper l'exécution il faut cliquer le bouton  la fin du programme n'étant pas prévue dans le code.

Conclusion

Dans cette séquence nous avons vu quelques programmes qui ne demandent pas la participation de l'utilisateur. On appel cela des "traitements par lots" ou "batch processing" en anglais.

Le déclenchement de tels traitement peut être planifié et automatisé.

A l'origine, les ordinateurs étaient alimentés en entrée par des instructions encodées sur des cartes perforées et fournissaient les données de sortie sur des imprimantes. On peut citer comme exemple de traitement par lots l'édition des fiches de paye dans une entreprise, mais aussi plus récemment l'envoi d'emails de "démarchage publicitaire"…

Ce n'est qu'à partir des années 1980, lorsque les ordinateurs sont devenus plus puissants et moins chers, que le mode interactif est apparu tel que nous le connaissons aujourd'hui. Un logiciel en mode interactif ou transactionnel n'est pas automatisé, il est exécuté par un utilisateur présent devant l'écran et se déroule en temps réel. C'est l'objet de la séquence suivante.

Yves Durasnel
Club sympaTIC
Octobre 2018

Inspiré du FUN Mooc "Introduction aux technologies des médias interactifs numériques"
par Pierre Cubaud et Stéphane Natkin du CNAM

2 Pour aller plus loin

Conseil médiation

Pour aller plus plus loin sur le sujet, nous vous conseillons de vous référer à la fiche outil “Initiation à la programmation“