Jean-Michel GÉRIDAN
Jean-Noël LAFARGUE

{Processing}

// S'initier à la programmation créative

DUNOD

Maquette de couverture : Misteratomic

5 rue Laromiguière, 75005 Paris
www.dunod.com

ISBN 9782100737840

Sommaire

Avant-propos

À qui s'adresse ce livre

Processing est un langage de programmation dédié à la production artistique, et notamment à la production d'images, ce qui explique sa large diffusion dans les écoles d'art et de design graphique et interactif. Grâce à ses nombreux modules additionnels, Processing ne se limite pas à la création visuelle et peut communiquer avec des dispositifs électroniques de type Arduino, avec des services Internet, peut manipuler du son, de la vidéo, etc. C'est un langage à la fois simple, puissant et bien conçu, appartenant à la famille de Java et de C++, qui constitue pour ces raisons une excellente initiation à la programmation informatique. Le logiciel Processing, qui sert à rédiger et à exécuter des programmes dans le langage du même nom est par ailleurs gratuit et disponible sur trois plates-formes : Mac OS, Windows et Linux.

Processing intéressera tout particulièrement les créateurs qui veulent produire des installations interactives à l'aide de périphériques répandus tels que la souris, le clavier ou la caméra, mais aussi à l'aide du capteur de mouvements Kinect, ou bien encore, en association avec une carte de prototypage Arduino, avec des capteurs de distance, de mouvement, de température, de localisation GPS, etc.

Processing permet aussi aux graphistes de générer des images ou des motifs à partir de données, ce qui en fait un langage très adapté à la « *data visualisation* » (appelée en français « graphisme de données » ou encore « graphisme d'information »), et lui vaut d'être employé pour ce genre d'usage par des institutions scientifiques qui souhaitent rendre intelligibles ou séduisant le résultat de leurs recherches.

Enfin, alors que l'on parle beaucoup d'enseigner la programmation informatique aux enfants dès l'école primaire, Processing peut être un outil de choix, du fait de sa simplicité d'emploi et de l'immédiateté des résultats qu'il produit.

Comment utiliser ce livre

Le livre que vous tenez dans les mains est à la fois un cours progressif et thématique, un ouvrage de référence qui passe en revue les principales fonctions du langage Processing, mais aussi un ouvrage d'initiation à la programmation informatique et à l'image numérique, pensé pour être accessible aux grands débutants comme aux programmeurs

affirmés – qui ont surtout besoin de comprendre la philosophie du langage et la liste de ses fonctions.

Si vous découvrez la programmation, la méthode la plus indiquée est de lire ce livre dans l'ordre de ses chapitres, devant un ordinateur où vous aurez ouvert le logiciel Processing et où vous saisirez et testerez le code : rien de plus parlant que de voir le résultat de ce que l'on a codé. Les programmes sont pour la plupart très courts, limitant les risques d'erreurs de saisie.

Certaines fonctions ne sont pas détaillées dans le livre, et il est possible que, au cours des versions à venir, de nouvelles fonctions apparaissent. Pour être toujours au fait des changements et bénéficier d'une référence complète du langage Processing, pensez à consulter l'aide depuis le menu Help > Reference, qui permet d'accéder à une liste exhaustive du vocabulaire de Processing et de comprendre les effets de chaque commande grâce à de courts exemples.

À propos des auteurs

Jean-Michel Géridan a reçu une formation à l'École supérieure d'art et de design de Reims, à l'université Paris 8 et à l'École nationale supérieure des arts décoratifs. Spécialisé dans le design graphique et les nouveaux médias, cofondateur de la maison d'éditions Franciscopolis, il a enseigné plusieurs années à l'École d'art du Havre avant de devenir directeur de l'École supérieure d'art de Cambrai.

Jean-Noël Lafargue, formé à la peinture aux Beaux-Arts de Paris et à la réalisation multimédia à l'université Paris 8, est ou a été enseignant dans plusieurs écoles d'art françaises : Amiens, Rennes, Angoulême et Le Havre, notamment. Il est l'auteur de plusieurs livres de sujets divers : bande dessinée, technologies, histoire culturelle des mythes apocalyptiques, la science-fiction. Ses sujets d'intérêt se trouvent évoqués sur ses divers blogs, accessibles à l'adresse http://www.hyperbate.fr.

Ensemble, Jean-Michel Géridan et Jean-Noël Lafargue ont coordonné l'équipe de recherche IDeA (Interactivité, design et art) et rédigé deux livres : *Processing, le code informatique comme outil de création* (Pearson, 2011) et, associés à Bruno Affagard, *Projets créatifs avec Arduino* (Pearson, 2014).

Remerciements

Les auteurs remercient en premier lieu leurs étudiants ainsi que les lecteurs de leurs ouvrages précédents pour leurs remarques et leurs requêtes diverses, grâce auxquelles ils ont pu apporter de multiples améliorations au présent ouvrage.

Ils tiennent à saluer amicalement la nombreuse communauté qui fait vivre le langage Processing en France, notamment dans les écoles d'art et de design : Douglas Edric Stanley, Jeff Guess, Yannick Mathey, Caroline Kassimo-Zahnd, Sylvie Tissot, Dominique Cunin, Alexis Chazard, Olivier Cornet, Benoît Wimart, Bruno Affagard, Loïc Horellou, Bachir Soussi-Chiadmi, Antonin Fourneau, Mark Webster, Quentin Bréant, Julien Gachadoat, Antoine Schmitt, Uroš Petrevski, Fernand Dutilleux, Normals, Superscript, *n*-graphes…

Mais aussi : Vanina Pinter, Jean-Louis Boissier, Liliane Terrier, Claude Closky, Heiko Hansen, Véronique Marrier, Annick Lantenois, Gilles Rouffineau, Stéphane Trois carrés, Christelle Kirchstetter, Jérôme Saint-Loubert Bié, Juliette Pollet, Barbara Dennys, Thierry Heynen, Yann Owens, Keyvane Alinaghi, Jean-Louis Fréchin, Étienne Mineur,

Peter Gabor, Laure Limongi, Anne Zeitz, Stéphanie Solinas, Alexandre Laumonier, Marie Lechner, Julie Morel, Stéphane Degoutin, Gwenola Wagon.

Enfin, nous remercions les éditions Pearson, chez qui nous avons publié un livre sur le même sujet il y a cinq ans, de nous avoir autorisés à faire celui-ci.

1 Premier contact

L'ORIGINE DU PROJET PROCESSING

Processing a été initié au printemps 2001 par Ben Fry et Casey Reas, deux étudiants du *Aesthetic and Computation research group* du Media Lab du MIT. Leur logiciel reprenait un peu la philosophie minimaliste de *Design by numbers*, un logiciel de création visuelle par le code informatique créé deux ans plus tôt par John Maeda, leur professeur, lui-même influencé par Muriel Cooper, graphiste du MIT qui y a enseigné le graphisme algorithmique de 1973 à 1994, l'année de son décès. *Design by numbers* n'a jamais été autre chose qu'un outil pédagogique pour enseigner les principes de l'image numérique. **Processing**, en revanche, a toujours eu l'ambition d'être un logiciel de production dans le domaine de la création visuelle et interactive sur supports numériques, en concurrence avec des logiciels propriétaires tels que Director, Flash, Max/MSP. Testé dans de nombreuses écoles d'art et de design, Processing est longtemps resté en version « beta », c'est-à-dire en version expérimentale. En 2005, le logiciel a reçu un prix au festival Ars Electronica. En 2008, devenu mûr, Processing est enfin passé à sa première version de production, la 1.0. Cette même année, John Resig, l'auteur du framework JQuery, a commencé à travailler à **Processing.js**, un portage de Processing pour les navigateurs web.

En 2012, Ben Fry, Casey Reas et Daniel Shiffman ont fondé la fondation Processing, qui pilote l'évolution de Processing, Processing.py et P5js et perçoit les dons des utilisateurs – dons qui consistuent l'unique source de financement pour le développement de Processing, créé dans un esprit de gratuité et de partage.

Plus qu'un simple langage de programmation, Processing constitue une philosophie de création, et sert de modèle à de nombreux projets : Wiring, Arduino, Android Processing, Processing.py, P5js, Cinder ou encore openFrameworks. Des versions Scala, Lisp ou Ruby de Processing sont en cours de développement.

Un certain nombre d'artistes ou de graphistes utilisent Processing comme outil de production : Ben Fry et Casey Reas, bien entendu, mais aussi Robert Hodgin, Golan Levin, Toxi, Andreas Gysin, Marius Watz, Aaron Koblin, George Legrady, HeHe, Superscript2, Joshua Davis…

TÉLÉCHARGEMENT ET INSTALLATION

Processing est diffusé sous forme de logiciel « libre », ce qui implique entre autres que son acquisition et son utilisation sont gratuites pour tous ses utilisateurs, et ceci sans limitation d'aucune sorte.

La première étape pour installer Processing est de télécharger le logiciel à l'adresse :

http://www.processing.org/download

La figure 1.1 montre les cinq versions disponibles : une version pour le système Mac OS X, deux versions pour le système Linux et deux versions pour le système d'exploitation Windows.

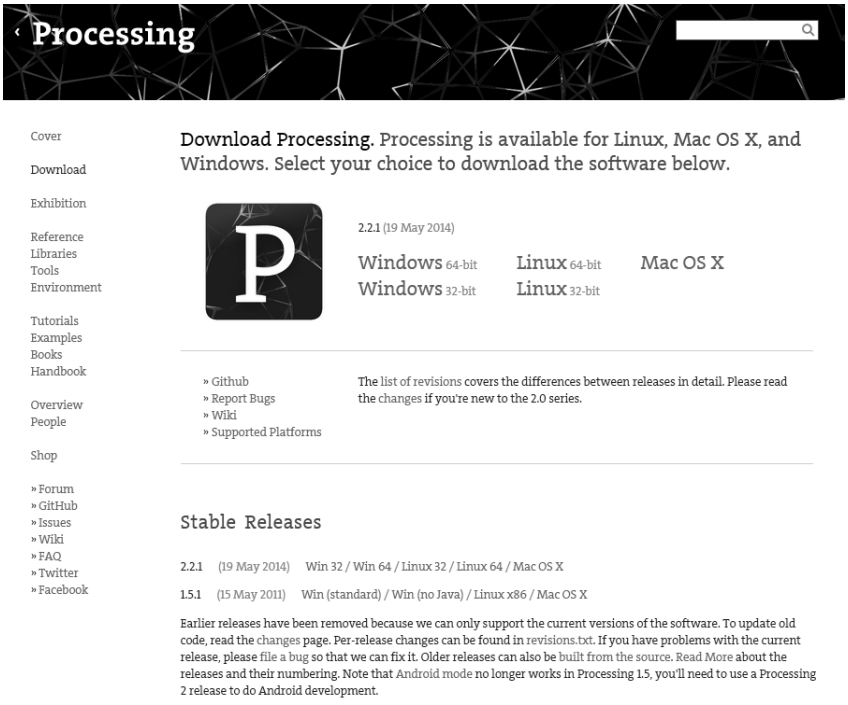


Figure 1.1 Les versions de Processing disponibles

Avant d'accéder aux différents liens de téléchargement, le site propose à ceux qui le souhaitent de faire un don à la fondation Processing, qui accompagne le développement du logiciel et du langage. Ce don est bien entendu tout à fait facultatif, il suffit de cocher la case « No Donation » pour passer à l'étape suivante.

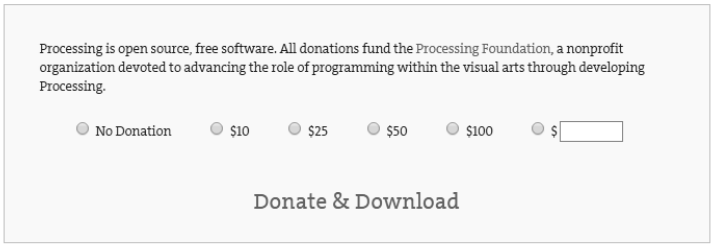


Figure 1.2

Les cinq versions de Processing pouvant être téléchargées :

- **Linux 32 bits** et **Linux 64 bits** – Le fichier téléchargé est une archive au format .tgz, que l'on peut décompacter et que l'on peut ensuite placer où l'on veut sur son ordinateur, selon les droits d'administration dont on dispose sur son système et selon la manière dont on l'a organisé. La plupart des gens l'installent dans leur dossier utilisateur. On choisit d'installer la version 32 bits ou la version 64 bits selon la version du système d'exploitation que l'on utilise.
- **Mac OS X** – Sous le système Mac OS X, le fichier téléchargé (une image-disque au format compressé .dmg, comme la plupart des logiciels à télécharger pour Macintosh) arrive typiquement sur le bureau de l'ordinateur. Il s'exécute généralement de manière automatique. S'il ne le fait pas, il suffit de double-cliquer sur l'icône du fichier. Lorsque ce fichier est décompacté, une fenêtre apparaît et suggère de glisser le dossier Processing dans le dossier « Applications » du système. Les versions de Processing supérieures à Processing 2.0 imposent au minimum l'usage de Mac OS X 10.8.3.
- **Windows 32 bits** et **Windows 64 bits** – Sous Windows (Windows 7 et au-dessus), le fichier téléchargé est une archive au format .zip, que l'on peut décompacter à l'aide d'un outil approprié (Winzip, Winrar…) mais que Windows sait aussi traiter tout seul. Cette archive contient un sous-dossier qui contient lui-même des sous-dossiers. L'ensemble doit être placé sur l'ordinateur, de préférence dans le dossier « Program files » de Windows. On choisit la version 32 bits ou la version 64 bits du système utilisé. Pour connaître la version que l'on a installée, il faut vérifier les informations générales du système, situées dans Panneau de configuration > Système et sécurité > Système.

Quel que soit le système employé, Processing peut être exécuté sans être installé – il n'est pas fourni avec un programme d'installation et fonctionne sans avoir besoin de modifier le système d'exploitation. Il est cependant avisé de ranger le programme dans un dossier approprié sur l'ordinateur. Sur Macintosh, ce n'est pas difficile, puisque Processing est constitué, en apparence, d'un unique fichier. Sur les autres plates-formes, Processing est constitué d'un dossier, qui lui-même contient des fichiers et des sous-dossiers qu'il faudra toujours déplacer ensemble afin d'en respecter l'arborescence.

Sur la page de téléchargement, on peut se procurer les toutes dernières versions de Processing, mais aussi la version 1.5.1, sortie en mai 2011, qui est, parmi les versions « stables » celle qui est la plus compatible avec d'anciens systèmes d'exploitation ou avec certaines librairies externes.

On peut trouver les anciennes versions de Processing, y compris versions alpha (pré-tests) et beta (tests viables) et le code source du logiciel sur la plate-forme Github :

https://github.com/processing/processing

L'ENVIRONNEMENT DE TRAVAIL

Processing est à la fois un langage et un environnement de travail.

Il est possible d'employer d'autres interfaces que le logiciel Processing pour programmer dans le langage Processing, par exemple le logiciel Eclipse. Nous nous limiterons à la méthode la plus courante, qui est d'utiliser le logiciel Processing. Notons que l'interface de Processing peut être utilisée pour manipuler d'autres langages que Processing, comme ses proches cousins Processing.js et Processing pour Android, mais aussi, comme le langage Python, très populaire auprès des chercheurs, notamment. Le passage d'un langage à un autre se fait par le biais des « modes », dont il sera question plus loin dans le livre.

Lorsqu'on lance Processing, on obtient une fenêtre assez simple.

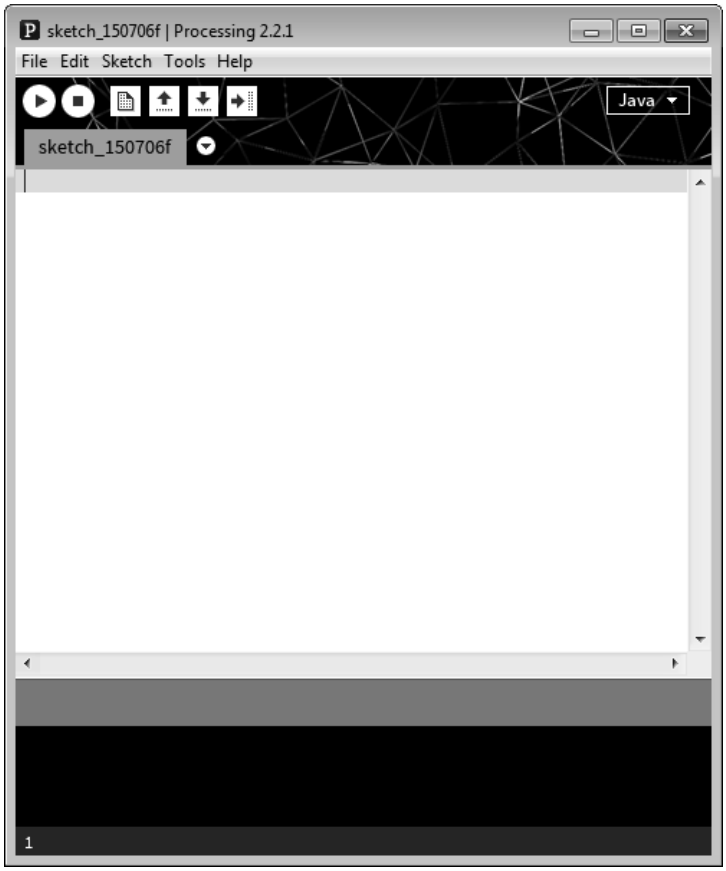


Figure 1.3 La fenêtre de démarrage de Processing

La plus grande partie de cette fenêtre est sa zone d'édition de texte, sur fond blanc. C'est à cet endroit que l'on rédige les programmes.

En dessous de cette zone se trouve une seconde zone de texte, sur fond noir, que l'on ne peut pas éditer soi-même et où s'affichent divers messages relatifs au fonctionnement du programme, les messages d'erreur, notamment. Notez que dans la version 3 de Processing, qui n'est pas encore terminée à ce jour, la zone de messages distingue messages et erreurs, on passe d'un type de message à un autre en cliquant sur le bouton correspondant.

Au-dessus de la zone d'édition des programmes nous trouvons une barre d'icônes (figure 1.3).

Le premier bouton en partant de la droite vous semblera très familier puisqu'il s'agit du pictogramme Play, qui est universellement employé dans la Hifi et, bien sûr, par divers logiciels permettant de lire un flux audio ou vidéo. En cliquant dessus, on lance l'exécution du code qui a été rédigé. Il est possible d'obtenir le même résultat en sélectionnant la commande « Run » du menu « Sketch », ou en utilisant la combinaison de touches Ctrl+R.

Si vous effectuez une de ces actions destinées à lancer l'exécution du programme, vous constaterez l'apparition d'une fenêtre de petite taille (100 × 100 pixels, précisément) et de couleur grise.

Si vous faites l'essai et que vous obtenez cette fenêtre grise (figure 1.4), ça signifiera que Processing fonctionne bien sur votre système d'exploitation.

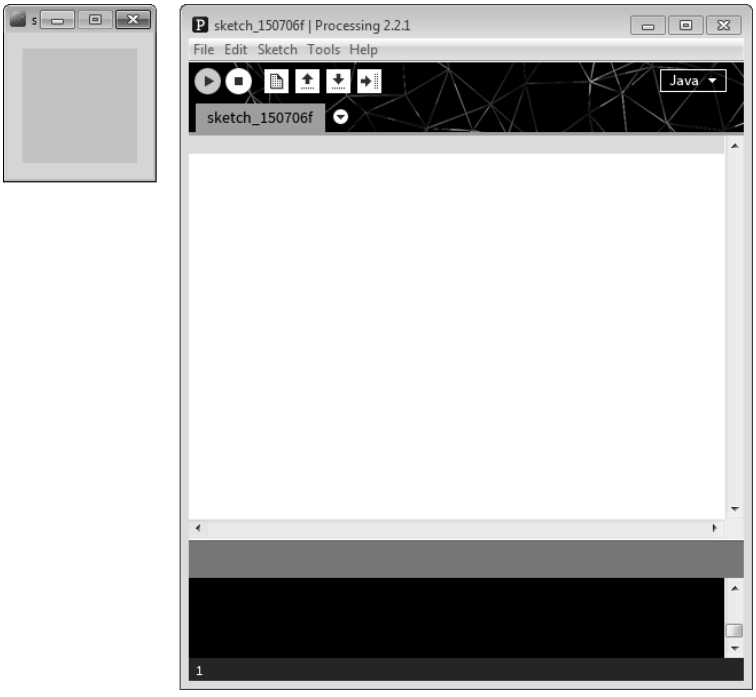


Figure 1.4 La fenêtre par défaut du programme.

Cette fenêtre peut-être fermée, comme toutes les fenêtres système, à l'aide de l'icône qui se trouve dans son bord supérieur droit. Mais on peut aussi utiliser le bouton Stop de Processing, parfois plus efficace, notamment lorsque la fenêtre d'exécution du programme ne répond pas correctement.

Figure 1.5 Les icônes Play, Stop, New, Open, Save, Export.

Les quatre icônes de la figure 1.5 permettent d'accéder à des fonctions essentielles :

- **New** – Nouveau programme.
- **Open** – Ouvrir un programme déjà enregistré.
- **Save** – Enregistrer le programme en cours.
- **Export application** – Convertir le programme en cours en application autonome.

Deux zones de « sortie » (*output*) ont été prévues à l'adresse des utilisateurs :

- une zone de message, en gris, où Processing affiche divers avertissements et commentaires ;
- une zone de sortie, en noir, où Processing affiche notamment les messages Java (en orange sur noir), qui sont souvent la version plus complète (mais moins lisible) des avertissements affichés sur la zone de message. Cette zone de sortie sert aussi à afficher des messages appelés depuis le programme à l'aide des commandes `print()` et `println()`.

Exemple

Si vous écrivez ce programme :

```
print("Bonjour à tous !");
```

et que vous l'exécutez en cliquant sur le bouton Play, vous verrez la chaîne de caractères `"Bonjour à tous !"` s'afficher, sans guillemets, dans la zone noire.

La commande `print()` est très pratique lorsque l'on veut connaître l'état d'une variable, notamment.

La commande qui suit, dont nous détaillerons le fonctionnement précis à la section 12.1, affiche dans la zone de sortie un nombre au hasard compris entre `-100.0` et `100.0` :

```
print(random(-100, 100));
```

À chaque exécution du programme, le nombre créé sera différent, et donc ce qui s'affichera dans la zone dédiée aussi.

La différence entre `print()` et `println()` est que la seconde commande saute une ligne avant d'afficher un message puis saute une autre ligne après avoir affiché ce message.

Exemple

Si nous lançons ce programme :

```
print("bonjour à tous");
print("bonjour à tous");
```

... ce qui apparaîtra dans la zone inférieure sera le message :

```
"bonjour à tousbonjour à tous"
```

sans saut de ligne et sans espace entre les deux phrases.

Si nous écrivons :

```
println("bonjour à tous");
println("bonjour à tous");
```

nous obtiendrons en sortie :

```
"bonjour à tous
bonjour à tous"
```

On peut utiliser le signe + pour concaténer (c'est-à-dire associer) des messages et des variables, notamment pour les afficher avec `print()` et `println()`, dans le but d'obtenir des chaînes de caractères, comme ceci :

```
int age = int(random(7, 77));
println( "Tintin convient aux personnes de "+age+" ans");
```

Ici, Processing a d'abord créé au hasard (*random*) un nombre compris entre 7 et 77, et l'a stocké dans la variable nommée `age`, puis a créé une chaîne de caractères composée de `"Tintin convient aux personnes de"`, de la valeur de `age` et enfin, de la chaîne de caractères `" ans"`.

Nous verrons dans le chapitre 4 ce que signifie le mot « variable », et dans le chapitre 12, comment on manipule le hasard.

Vous pouvez lancer le programme plusieurs fois de suite pour en voir les effets.

En haut à droite de l'environnement de travail de Processing se trouve un bouton en forme de flèche.

Lorsque l'on appuie dessus, un menu apparaît pour nous proposer d'ajouter des pages à nos programmes (New tab), de supprimer des pages (Delete), de renommer une page (Rename) et de naviguer parmi les différentes pages du programme (Previous/Next).

Si nous cliquons sur la commande New tab, Processing commencera par nous demander de nommer la page que nous créons, puis ajoutera une page vierge au programme. Dans la zone supérieure, nous aurons donc deux onglets, l'un portera le nom de notre programme principal et l'autre, celui de la nouvelle page que nous aurons créée. Il sera possible de naviguer d'une page à l'autre en cliquant sur les onglets.

Notre « *sketch folder* » (le dossier dans lequel sont stockés les fichiers de notre programme) contiendra désormais deux fichiers `"pde"`.

Au moment de l'exécution, Processing traite toutes les différentes pages de programme d'un même dossier comme un seul et unique programme. L'intérêt de cette division du programme en plusieurs parties est de gagner en lisibilité et en organisation. Cela se révèle particulièrement utile pour la programmation objet, par exemple.

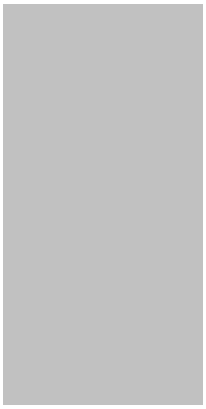
Puisque toutes les pages du programme sont traitées comme une seule, celles-ci ne doivent pas contenir de doublons : il n'est pas possible, par exemple, d'avoir deux fonctions `draw()` différentes dans les deux pages d'un même programme. Par ailleurs, dans le cadre de la programmation objet, les pages de programme ne doivent jamais avoir les noms des classes qu'elles contiennent.

UNE TOUTE PREMIÈRE APPROCHE

Nous vous proposons de prendre un premier contact avec Processing en effectuant pas à pas la petite expérience qui suit.

Nous avons déjà vu que, lorsque nous cliquons sur le bouton « Play », en l'absence de tout programme, une fenêtre grise apparaît. Cette fenêtre mesure 100 pixels de large par 100 pixels de haut.

Dans la zone d'édition de texte du logiciel Processing (zone blanche), écrivez à présent cette ligne :



```
size (100, 200);
```

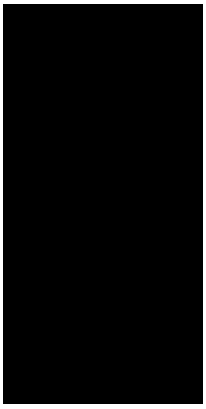
Notez bien le point-virgule qui se trouve à la fin de la ligne. S'il est omis, le programme ne fonctionnera pas.

Les espaces n'ont pas d'importance, mais vous devez faire attention à respecter la casse (majuscules et minuscules) des caractères, car pour Processing, `size` est une commande précise qui ne sera pas compréhensible si l'on écrit « `SIZE` », « `Size` » ou « `sIZe` ».

Cliquez à nouveau sur le bouton Play.

Cette fois-ci, la fenêtre, remplie de gris, sera nettement plus grande en hauteur. Deux fois plus grande exactement, puisqu'elle mesurera 200 pixels de haut pour 100 de large.

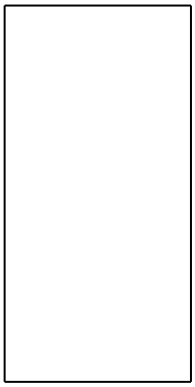
À présent, vous pouvez ajouter une seconde ligne de code à la zone d'édition, votre programme devra être celui-ci :



```
size (100, 200);
background(0, 0, 0);
```

Cette fois, après avoir cliqué sur le bouton Play, vous constaterez que la fenêtre a conservé ses nouvelles dimensions mais qu'elle est désormais noire et non plus grise.

Nous verrons plus tard le fonctionnement des couleurs, mais vous pouvez dès à présent tester le même code en modifiant les valeurs de la ligne qui concerne la couleur de la fenêtre.



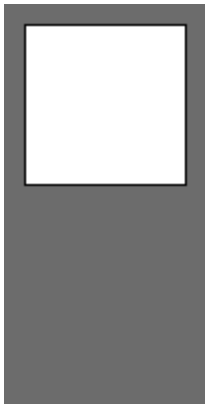
```
background(255, 255, 255);
```

```
background(127, 127, 127);
```

et ainsi de suite…

Notez que la valeur minimale de chaque chiffre séparé des autres par une virgule est 0 (zéro) et que sa valeur maximale est 255.

Pour finir, nous allons compléter notre programme d'une troisième commande, de cette manière :



```
size(100, 200);
background(127, 127, 127);
rect(10, 10, 80, 80);
```

Cette fois, en cliquant sur Play, un rectangle blanc cerné de noir et mesurant 80 × 80 pixels s'affichera au haut de l'écran.

Nous pouvons nous arrêter là, vous venez de rédiger votre tout premier programme en langage Processing.

PROCESSING RÉPOND-IL À VOS BESOINS ?

En feuilletant ce livre, vous serez peut-être effrayé en d'y voir beaucoup de chiffres, d'opérateurs, et de mots qui évoqueront chez vous quelques vieux souvenirs d'école : sinus, cosinus, angle... Peut-être faites-vous partie des gens qui pensent que la création artistique est l'ennemie des mathématiques et réciproquement.

Vous verrez pourtant qu'aucune notion abordée ici n'est particulièrement complexe, et que le niveau requis en mathématiques ne dépasse pas celui de la classe de troisième, au pire. Les auteurs de ce modeste manuel sont tous deux d'anciens étudiants en école d'art et non des ingénieurs. Vous n'aurez pas non plus besoin de mémoriser une grande quantité de fonctions et de mots-clés.

En fait, la programmation telle que nous la pratiquerons ici réclame avant tout un peu de logique et la compréhension de quelques principes.

Le fonctionnement assez « sans façons » de Processing permettra de votre part un apprentissage très progressif. Pour cette raison, et selon notre expérience d'enseignants, nous pouvons dire que, quel que soit votre « bagage » en mathématiques et en informatique, vous pouvez très aisément vous mettre à la programmation avec Processing.

Si vous êtes graphiste ou artiste et que vous souhaitez réaliser des images ou des animations géométriques interactives ou non, si vous voulez traiter des signaux (transformer les images reçues par une webcam par exemple), si vous souhaitez intervenir sur du son, si vous vous intéressez au graphisme d'information, si vous êtes un scientifique à la recherche d'un outil simple pour réaliser des simulations, alors il est probable que Processing soit fait pour vous.

QUELQUES RÈGLES DE MISE EN FORME DU CODE

Avant de commencer à travailler, nous devons effectuer une mise au point au sujet de la rédaction du code en langage Processing. Les règles à respecter ne sont ni nombreuses, ni illogiques, ni complexes à retenir, mais il est important de les connaître car elles sont la première cause d'erreurs chez les débutants. On les intègre cependant assez vite.

Sensibilité à la casse

Nous avons vu précédemment que Processing était sensible à la casse des caractères : on ne peut pas écrire `"siZe"` à la place de `"size"`, ce seront, pour Processing, deux mots différents. Nul besoin de le dire, mais Processing est aussi sensible à l'orthographe. En effet, `"background()"`, `"width"` ou `"height"` sont des mots qui existent dans Processing, mais « bakground() », « widht » et « heigh » ne veulent rien dire.

> On remarquera que l'éditeur de code de Processing colore les mots, afin d'aider le rédacteur du code à repérer facilement ses propres fautes de frappe. Les commandes sont ☞

☞

colorées en bleu ; les types de variables en rouge ; etc. Malheureusement, cette fonction fort pratique ne signale pas toujours les erreurs de majuscules/minuscules.

Utilisation des espaces

Les espaces servent à séparer les mots. Processing n'est pas troublé lorsque l'on place plusieurs espaces. Les deux lignes de code qui suivent sont équivalentes :

```
print("bonjour") ;
print   (   "bonjour"    )   ;
```

Utilisation des sauts de ligne

Dans Processing, les sauts de ligne ont une utilité cosmétique, excepté dans le cas des commentaires (voir plus loin). Cela signifie que l'on peut ne pas sauter de ligne entre des commandes, ou au contraire, en passer plusieurs. Les commandes ne sont pas séparées par des sauts de ligne, mais par le point-virgule, comme dans la ligne qui suit, où trois commandes sont placées à la suite les unes des autres :

```
size(500,500) ; print("ok") ; fill(255) ;
```

Pour ordonner proprement son code, on peut recourir à la commande `"auto format"`, qui se trouve dans le menu « edit » de Processing : cette commande ajoute des sauts de ligne et des indentations au code, dans un souci de lisibilité et d'uniformité.

Ce qui est ouvert doit être fermé

Plusieurs caractères servent à délimiter des sections :

- `"` : les doubles guillemets servent à encadrer les chaînes de caractères, par exemple `"ceci est une chaîne de caractères"`.
- `'` : les simples guillemets servent à encadrer les caractères simples, par exemple `'a'`.
- `(` et `)` : les parenthèses servent à encadrer des expressions numériques ou logiques. Leur utilité est la même qu'en mathématiques, où (1 + 4) * (2 * 2) donne 20, tandis que 1 + 4 * 2 * 2 donne 17.
- `{` et `}` : les accolades servent à encadrer les blocs d'instruction. On est forcé d'en utiliser lorsque l'on rédige des fonctions, et ils servent aussi pour les structures de contrôle et les itérations. Sur Macintosh, on les obtient grâce aux combinaisons de touches alt+(et alt+). Sur PC, on les trouve sur le clavier.
- `[` et `]` : les crochets servent à accéder à un élément précis dans un tableau. Sur Macintosh, on obtient ces caractères avec les combinaisons alt+control+(et alt+control+). Sur PC, on les trouve sur le clavier.
- `/*` et `*/` : section de commentaires (voir la section suivante).

Nous reverrons chacun de ces éléments en temps voulu, mais il est important de retenir que lorsque l'on ouvre une délimitation avec l'un de ces caractères, celle-ci devra forcément être fermée. Et si on ne le fait pas, le programme renverra forcément un message d'erreur au moment de son exécution.

La ligne qui suit a pour effet d'écrire `"bonjour"`. Sa formulation est inutilement complexe, mais elle est juste, car chaque délimitation ouverte finit par être fermée :

```
{{}{{print(("bonjou"+'r')));}}}
```

en revanche, celle qui suit renverra un message d'erreur :

```
{{{{print(("bonjou"+'r')));}}}
```

...on comprend pourquoi en comptant le nombre d'accolades ouvertes (4), qui est différent du nombre d'accolades fermées (3). Avec un peu d'expérience, on évite sans peine ce genre d'erreurs.

Les commentaires

Finissons avec le sujet de la mise en forme du code par une notion très importante : les commentaires. Ceux-ci servent à inhiber une partie du programme, soit dans le but de l'anoter avec des phrases qui ne peuvent pas être exécutées, soit dans celui de rendre inactives des lignes dont on veut temporairement tester l'absence.

Il existe deux syntaxes différentes, correspondant aux commentaires qui tiennent sur une seule ligne, et qui suivent une paire de double slashs (`//`), et aux sections de commentaires qui tiennent sur plusieurs lignes et sont encadrés par les caractères slash-astérisque (`/*`) en entrée et astérisque-slash (`*/`) en sortie.

```
/* cette partie du programme, qui est concentrée entre les carac-
tères slash-astérisque et astérisque-slash ne sera pas exécutée par
le programme. */


// cette partie du code ne sera pas exécutée non plus
```

2 Géométrie I

COORDONNÉES

Dans la section suivante, nous allons voir comment dessiner des formes simples telles que le point, la ligne, le triangle, le rectangle, le quadrilatère quelconque et l'ellipse. Avant d'étudier chacune de ces formes et les commandes qui permettent de les dessiner, nous devons faire le point sur le système de coordonnées de la fenêtre d'exécution de nos programmes.

Il s'agit d'un système de coordonnées cartésiennes, qui diffère du modèle généralement employé en mathématiques par la position de son point d'origine.

En effet, l'origine (le point 0,0) de notre fenêtre d'affichage se trouve ici située à son coin supérieur gauche : la progression des abscisses (l'axe horizontal) va de gauche à droite, comme c'est le cas du modèle habituellement utilisé en mathématiques, mais la progression des ordonnées va de haut en bas, et non le contraire. De gauche à droite et de haut en bas : c'est, finalement, le sens de lecture occidental.

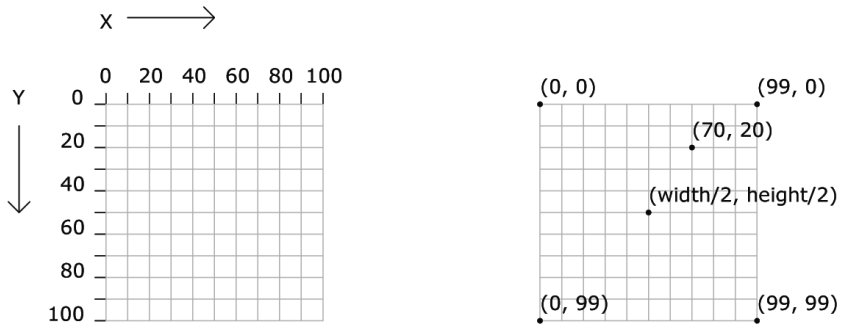


Figure 2.1 Système des coordonnées sur Processing

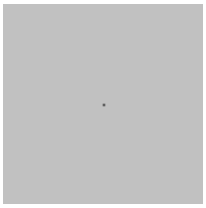
L'unité employée ici est le pixel (*picture element*), c'est-à-dire le plus petit point que notre ordinateur sait afficher.

Chaque dispositif d'affichage (écran, vidéoprojecteur) a un format différent : 1 024 × 768, 1 366 × 768, 1 920 × 1 200, etc. **Par convention, on dit que les écrans ont une résolution de 72 pixels par pouce (1 × 1 pouce, soit 2,55 × 2,55 centimètres, correspondrait à 72 × 72 pixels).** Mais cette convention a peu de sens sur écran, et n'a d'intérêt que dans le domaine de l'imprimé.

FIGURES SIMPLES

point()

Le point est composé de deux éléments, à savoir une abscisse (*x*) et une ordonnée (*y*). Pour dessiner un point à l'exact milieu de notre fenêtre, nous écrirons ceci :

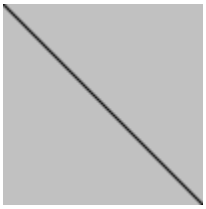


```
point(width/2, height/2);
```

`width/2` signifie la largeur de l'écran divisée par deux, et `height//2` la hauteur de l'écran divisée par deux.

line()

La commande qui sert à dessiner un segment de droite s'invoque avec quatre arguments, qui correspondent aux coordonnées de deux points, un point « A » (*x*1,*y*1) et point « B » (*x*2,*y*2). Pour dessiner un segment de droite partant du point en haut à gauche de la fenêtre (`0,0`) et arrivant en bas à droite (`width,height`), on écrira :



```
line(0,0,width,height);
```

Notez qu'écrire `line(x1,y1,x2,y2)` est équivalent à écrire `line(x2,y2,x1,y1)`.

triangle()

Un triangle est composé de trois segments de droite qui relient trois points. La commande `triangle()` s'invoque donc avec six arguments, qui sont en fait trois paires de coordonnées :

```
triangle(x1,y1,x2,y2,x3,y3)
```

```
// premier triangle
triangle(0,80,40,0,80,80);
// second triangle
// (qui recouvrera le premier,
// puisqu'il est dessiné après)
triangle(20,100,60,20,100,100);
```