TP 2 – Initiation à Processing.org

Ph. Truillet

Septembre 2021 – v. 1.1

1. introduction

Processing (**http://processing.org**) est un langage (en réalité, une **surcouche** du langage Java) développé par Ben Fry et Casey Reas (MIT) et dédié à la programmation créative.
Il permet entre autres de manipuler des images (fixes ou animées) et interagir avec elles. Processing permet aussi de générer des textes, des formes vectorielles, des images en trois dimensions (en utilisant OpenGL), du son et plus généralement … **tout ce que vous souhaitez** !

Processing peut aussi être étendu à l'aide de librairies (fichiers JAR – **J**ava **AR**chive) ou grâce à des projets « *proches* » comme Arduino (projet directement dérivé de Processing) (`http://www.arduino.cc`) et Wiring (`http://wiring.org.co`) pour la gestion de capteurs physiques.

Il existe actuellement de multiples versions de Processing dont « ***p5.js*** » (`http://p5js.org`), « ***processing JS*** » (`http://processingjs.org`) écrits en javascript (reprenant la syntaxe de Processing), Processing Python (`http://py.processing.org`), Processing R, etc…

Figure 1 : Exemple de sketch écrit en Processing

Processing dans sa version originale s'appuie sur le langage Java, ce qui lui permet de fonctionner sur les systèmes d'exploitation Windows, MacOS X et Linux (et donc aussi sur Raspbian pour Raspberry Pi) et **Android** avec le mode idoine proposé avec Processing 3).
Malgré encore quelques petites lacunes (de plus en plus corrigées), Processing a de nombreux atouts dont notamment sa simplicité d'usage qui en fait un langage presque idéal pour le prototypage rapide !

2. installer Processing

Nota : Nous supposerons dans la suite du document que l'IDE (**I**ntegrated **D**evelopment **E**nvironment) Processing 3.5.4 est installé dans le répertoire `C:/langages`

Si ce n'était pas le cas, la première chose à faire est de se rendre à l'adresse `https://www.processing.org/download/?processing` et télécharger une des versions proposées (version stable 3.5.4 du 17 janvier 2020 en 32 ou 64 bits).

Une fois téléchargée, décompressez l'archive dans le répertoire de votre choix et lancez l'exécutable « *processing* ».
Déterminer ensuite le répertoire où seront sauvés vos programmes (*sketches*)
(cf. *File | Preferences*), choisissez l'emplacement dans le champ « *sketchbook location* » et appuyez sur ok (cf. Figure 2).

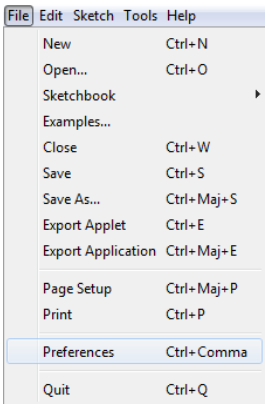


Figure 2 : Menu

Nota : Cette étape est primordiale pour la suite ; c'est dans ce répertoire que toutes librairies externes seront téléchargées et vos « sketchs » sauvés.

3. mon premier sketch

Il est possible d'utiliser Processing de différentes manières dont notamment en mode « *script* » (suites d'actions exécutées qu'une seule fois), et en mode « *continu* » (avec une boucle infinie – fonction ***draw***).

Le mode « *continu* » demande d'implémenter au moins deux fonctions : `setup()` qui initialise les variables (depuis Processing, on peut en plus utiliser la fonction `settings()`) et `draw()`, boucle d'affichage de données (*mainLoop*). Cette boucle permet d'afficher des animations graphiques complexes, réagir à des événements asynchrones provenant d'actions de l'utilisateur ou d'événements systèmes.

Par convention, les mots réservés du langage sont affichés en *bleu*, *vert* et *orange* dans l'IDE (cf. Figure 3).

Pour ce premier sketch, recopiez le code ci-après (vous trouverez le fichier source ici → `https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/primitives_graphiques.zip`) et lancez le script.

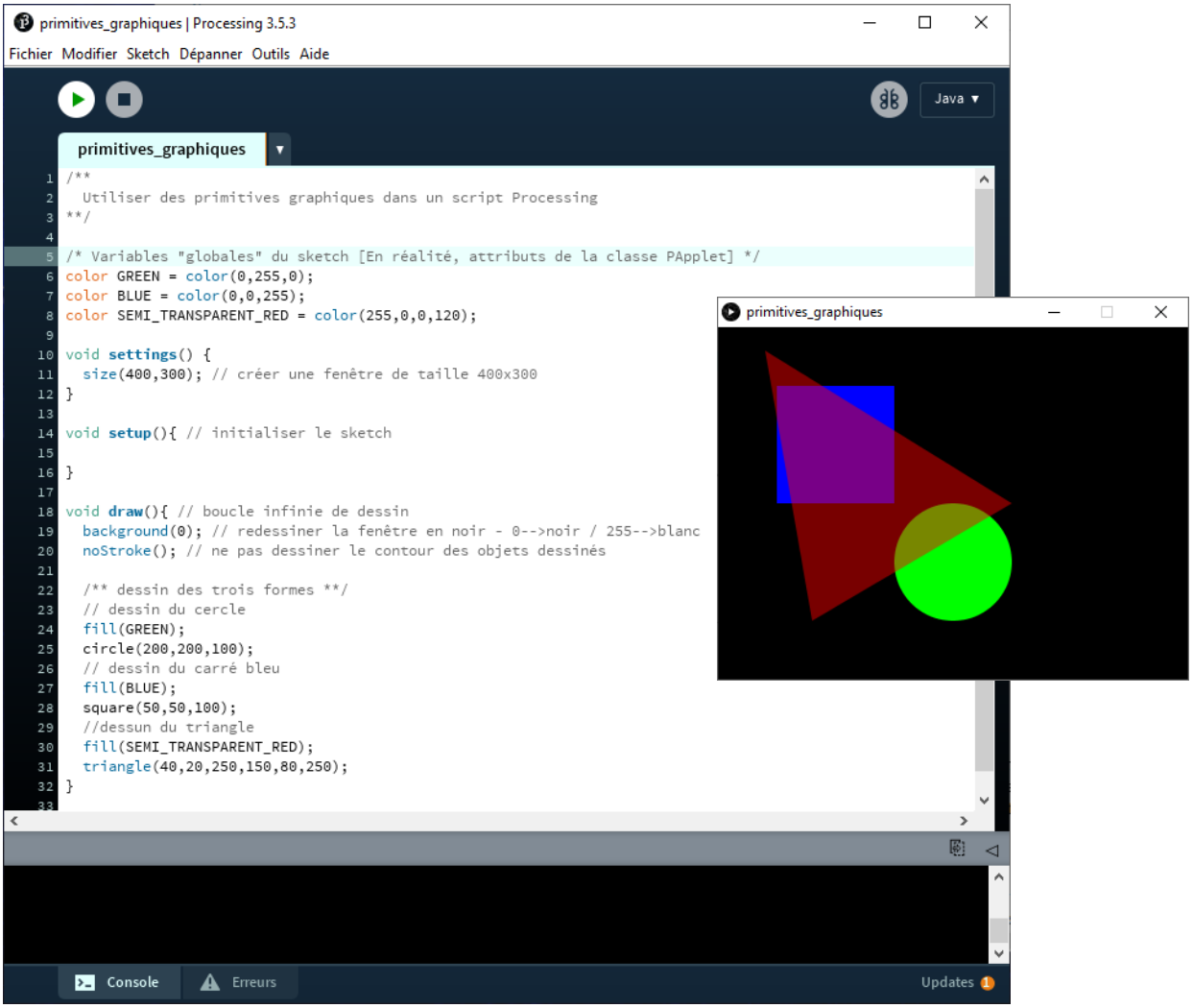
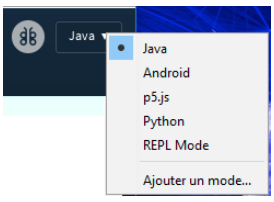


Figure 3 : mon premier sketch

Les possibilités du langage Processing sont quasi-infinies notamment avec la possibilité d'utiliser la programmation orientée-objet, d'ajouter des librairies externes et d'en écrire soi-même !

Enfin, l'IDE Processing propose plusieurs modes : Java (par défaut) mais aussi Android (ADB doit être installé), p5.js, Python, R et REPL (**R**ead **E**val **P**rint **L**oop soit sous forme de « *shell* »).



Il suffit de choisir le mode (une installation peut être requise) que vous souhaitez sur le bouton à droite de l'IDE.

Les premiers exercices ci-après vous donnerons un aperçu de ce qui peut être fait en quelques lignes de code.

4. formes & images

4.1 manipuler des formes

- Créer une nouvelle composition graphique en couleur (incluant de la transparence) utilisant au moins un morceau de cercle (arc), une ligne et un quadrilatère.
- Composer un tableau simplifié de type « *Mondrian* » (père du néo-plasticisme) ou Sophie Taeuber-Arp

  Voir des exemples ici : `https://www.wikiart.org/en/piet-mondrian`
  et là : `https://www.wikiart.org/en/sophie-taeuber-arp`

4.2 manipuler des images

- Afficher deux images ayant une teinte différente.
- Charger un fichier **png** avec un bit de transparence et créer une composition graphique en superposant les couches.
- Faire défiler de droite à gauche votre composition graphique.

4.3 utiliser de la typographie

- Afficher votre citation favorite avec votre police de caractères préférée.
- Utiliser deux polices de caractères différentes pour simuler un dialogue fictif entre 2 utilisateurs (phrases justifiées à gauche pour l'un et à droite pour l'autre)

5. un peu d'interaction

5.1 utiliser les entrées « classiques »

- Utiliser les flèches du clavier (droite, gauche, haut et bas) pour modifier la position d'un triangle affiché dans la fenêtre
- Dessiner un personnage qui réagit suivant les actions de la souris.

5.2 manipuler des films vidéo

Télécharger une vidéo sur Youtube au format mp4 (vous pouvez utiliser « *vlc portable* » ou par exemple le site `https://youtube-mp4.download/fr` pour la sauver)

Charger le projet **videos**
(`https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/videos.zip`) et modifier le code de telle manière que vous puissiez gérer le rembobinage, la pause et le démarrage de la vidéo avec les touches du clavier.

6. … et de mise en réseau

6.1 un flux RSS

Charger le projet **RSS** (`https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/rss.zip`), l'installer et l'ouvrir.
Exécuter le code.
Modifier le code de telle manière à télécharger le flux RSS (**R**eally **S**imple **I**ndication) du journal « *le Monde* » et afficher les « *Unes* » dans des cercles quand l'utilisateur clique sur le titre du journal.

7 capturer l'instant …

7.1 webcam

Installer au préalable la librairie **Video** disponible dans le menu *Outils* | *Ajouter un outil…* puis onglet *Libraries*.

Télécharger le projet webcam
(`https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/webcam.zip`)
Modifier le projet de telle manière qu'en appuyant sur la barre espace, on puisse modifier les effets à appliquer sur le flux vidéo (on souhaite notamment avoir comme effet : mode niveaux de gris et inverse vidéo)

7.2 "je suis … ton père ?!"

Installer au préalable la librairie **OpenCV** (**https://github.com/atduskgreg/opencv-processing/releases**) disponible dans le menu *Outils* | *Ajouter un outil…* puis onglet *Libraries*.

Modifier l'exemple *LiveCam* fourni par **OpenCV** et remplacer chaque figure détectée par une webcam par le masque de Dark Vador (**https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/darth_vader.png**)

8. de l'interaction avec « classe »

8.1 programmation orientée-objet

Télécharger le projet **interface** (https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/Gestion_Objets.zip) et modifier le code de telle manière **que l'objet change de couleur quand on clique dessus et revienne à sa couleur initiale quand on le relâche** (penser à utiliser les événements *MouseDragged()*, *MousePressed()* et *MouseReleased()*).

8.2 une machine « dans tous ses états »

Une manière efficace de concevoir des programmes interactifs est d'utiliser une machine à états. Le programme passe d'états en états grâce à des transitions qui sont la plupart du temps des événements provenant soit de l'utilisateur (actions de la souris, sur le clavier, …), soit du système lui-même.

La fonction **draw()** va finalement ressembler à la structure suivante, proposant différentes visualisations suivant l'état courant :

```
FSM mae // Machine A Etats

…

void draw() {
       switch(mae) {
              case INITIAL :
              …
              break;
       …
              default:
                     …
                     break;
       }
}
```

A partir de l'exemple fourni ici (https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/Machine_Etats.zip), **écrire** un programme qui **démarre la webcam / arrête la webcam quand l'utilisateur tape sur la barre espace**. (Par défaut, une image de renardeau (tout mignon) sera affichée à la place du flux vidéo)

9. la mise en réseau des données

9.1 les données JSON

Charger maintenant le projet **JSONWeather** (https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/JSONWeather.zip) l'installer et l'ouvrir.
Aller sur http://home.openweathermap.org/users/sign_up et créer un compte (*gratuit*) sur Open Weather Map. Après s'être connecté, recopier la clé API (**API key**).

Dans le code Processing, repérer la ligne (requête) où se trouve « **&APPID=** » et coller la clé.
Exécuter le code. Modifier le code de telle manière **à afficher une icône correspondante à la place de la description et ajouter un bouton permettant le rechargement de la météo pour une ville différente**.

10. un peu de Réalité Augmentée

10.1 à base de QRCode

A partir de l'exemple téléchargé ici :
`https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/QRCode.zip`
Créer une application **qui affiche un objet 3D sur le QR code détecté.**

Nota : cette application utilise la librairie **ZXing** (**`https://github.com/zxing/zxing`**)

Nota2 : Pour **générer un QR Code** depuis Processing, voir le code ici :
`https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/QRCode_Generator.zip`

10.2 à base de TopCodes

A partir de l'exemple téléchargé ici : `https://github.com/truillet/TopCode`
Créer une application **qui affiche des informations sur des objets physiques équipés de TopCodes repérés par une webcam quand l'utilisateur clique sur l'objet.**

Nota : cette application utilise la librairie **TopCodes-Tangible Objet Placement Codes** réécrite pour Processing (voir `http://users.eecs.northwestern.edu/~mhorn/topcodes` pour la version originale)

10.3 BoofCV

Installer au préalable la librairie **BoofCV for Processing** disponible dans le menu *Outils | Ajouter un outil…* puis onglet *Libraries*. BoofCV (`https://boofcv.org`) est un ensemble de fonctions de manipulation d'images.

Aller dans *Fichier | Exemples…* Dans la fenêtre ouverte (*Java Examples*), aller dans le sous-répertoire *Contributed Libraries | BoofCVforProcessing* ouvrez le sketch *Fiducials*. L'ouvrir et l'exécuter.
Modifier le code de telle manière **que quand le pattern fiduciaire est détecté devant la caméra, un texte codé apparait à l'écran sur la pattern.**

Fiducial à télécharger :
`https://github.com/truillet/upssitech/blob/master/SRI/1A/TP/fiducial_BoofCV.pdf` (voir la documentation ici : `http://boofcv.org/index.php?title=Tutorial_Fiducials`)

Nota : Vous pouvez aussi essayer les autres exemples (*TrackingObject* ou *RemovePerspective* par exemple 😉) intéressants pour d'autres sujets à traiter (comme le suivi d'objet en temps-réel ou le changement de perspective d'un objet).

10.4 NyARtoolkit

NyARToolkit (`https://nyatla.jp/nyartoolkit`) est une version modifiée de la célèbre librairie ARToolkit développée par l'université de Washington il y a une vingtaine d'année (`http://www.hitl.washington.edu/artoolkit`).

Télécharger la librairie NyARToolkit pour Processing ici : `https://github.com/nyatla/NyARToolkit-for-Processing`
Dézipper le fichier `nyar4psg.zip` et place le répertoire (nyar4pasg) dans le répertoire **`libraries`** emplacement de sketchbook (ex : `C:\dev\processing`). Relancer **Processing.org** pour que le changement soit pris en compte.

Aller dans *Fichier | Exemples…* Dans la fenêtre ouverte (*Java Examples*), aller dans le sous-répertoire *Contributed Libraries | nyar4psg* et ouvrez le sketch *multimarker*.
A partir de l'exemple, **développer une application qui permet de sélectionner une zone filmée par la webcam et prend une photo (sauvée en jpeg) quand on appuie sur la barre espace.**

Nota : si vous voulez ne pas utiliser de pattern ARToolkit, vous pouvez créer le vôtre à partir d'une image grâce à l'exemple *Fichier | Exemples…* Dans la fenêtre ouverte (*Java Examples*), aller dans le sous-répertoire *Contributed Libraries | nyar4psg* et ouvrez le sketch *nftFilesGen* (Natural Feature Tracker). Ouvrir ensuite dans le même répertoire le fichier *simpleNft* en le modifiant avec le motif que vous venez de créer (modifier la ligne 22)

Nota 2 : les patterns ARToolkit kanji et hiro sont téléchargeables ici :
`https://niebert.github.io/SamplesAR/markers/Hiro_Kanji_3Markers.pdf`

11. un peu de distribution et de multimodalité

11.1 La classe Robot

La classe Robot de java (`import java.awt.Robot`) permet de prendre la main sur l'ensemble du bureau (Windows, MacOS ou Linux) y compris hors de la fenêtre en simulant les appuis souris et/ou au clavier.
Ecrire un **sketch Processing qui permet de prendre un screenshot de l'écran et le sauvegarder au format jpeg**.

11.2 Le bus logiciel ivy

Télécharger l'exemple ici
`https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/ivyP5.zip`

Dézipper les deux sketchs et lancer-les tous les deux. Ces deux sketchs utilisent **le middleware ivy** pour communiquer sur le réseau local (voir `http://www.eei.cena.fr/products/ivy` pour une information générale).

En cliquant sur la première fenêtre (sketch « *sender* »), un message sera affiché sur l'autre fenêtre (« *receiver* ») et un feedback est envoyé à la première.

Une fois compris le principe, **écrire une interface composée de plusieurs sketchs** (qui peuvent communiquer sur le réseau local) qui décode un QR-code présenté devant une caméra, affiche l'information décodée dans une autre fenêtre (d'une autre machine par exemple) et lit via une synthèse vocale le texte décodé (on pourra utiliser la librairie **ttslib** par exemple).

12. pour finir de manière un peu HARD…(ware)

Utiliser au préalable l'IDE Arduino (**http://www.arduino.cc**) sur votre machine.

Télécharger l'exemple ici :
`https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/processing_arduino.zip`

Compiler et téléverser le code **capteur.ino** sur le module arduino branché sur le port série. Brancher une led infrarouge sur le pin Analogique **A0** et **GND**. (le code va lire la valeur du capteur et l'écrire sur le port série)

Exécuter le code Processing.
Modifier le **code de telle manière que la valeur du capteur récupérée soit affichée sous forme de barre verticale entre 0 (si la valeur récupérée est « *0* ») et 300 pixels maximum** (si la valeur récupérée est « *1024* »)

13. adresses utiles

- **Processing :** `http://www.processing.org`
- **P5.js :** `http://p5js.org`

- **Référence :** `http://processing.org/reference`
- **Learning Processing :** `http://www.learningprocessing.com`
- **Hello Processing :** `http://hello.processing.org`

- **Support de cours :**
  `https://github.com/truillet/upssitech/blob/master/SRI/1A/Cours/C_processing.org_2.2.pdf`
- **Librairies :** `https://processing.org/reference/libraries/`