Processing.org

Ph. Truillet

Janvier 2019 – v. 2.61

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 créer des images (fixes ou animées) et interagir avec elles. Processing permet aussi de manipuler, générer des textes, des formes vectorielles, des images en trois dimensions (en utilisant OpenGL), du son, … et plus généralement tout !
Processing peut enfin être étendu à l'aide de librairies ou grâce à des projets « *proches* » comme Arduino (dérivé de Processing) (**http://www.arduino.cc**) et Wiring (**http://wiring.org.co**) pour la gestion de capteurs

Il existe 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…

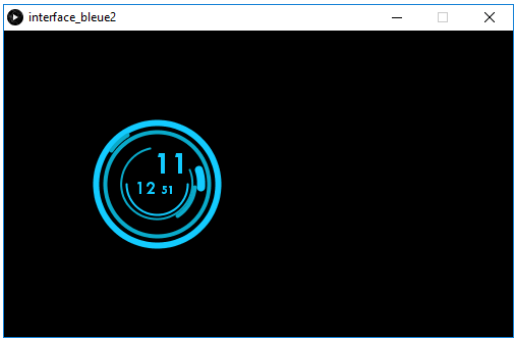


Figure 1 : Exemple de sketch écrit en Processing

Processing s'appuie sur le langage Java, ce qui lui permet de fonctionner sur les systèmes d'exploitation Windows, MacOS X et Linux (dont Raspbian sur Raspberry Pi) et **Android** avec le mode idoine proposé avec Processing 3).

Malgré quelques 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 Processing 3.4 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 3.4 du 26 juillet 2018 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).

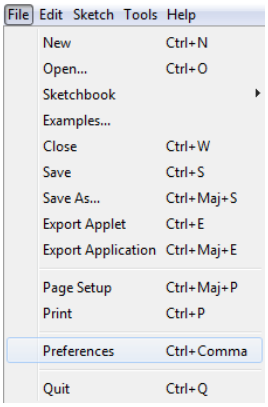


Figure 2 : Menu

3. mon premier sketch

Il est possible d'utiliser Processing de différentes manières dont notamment en mode « *script* » (suites d'actions exécutées 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 et **draw()**, boucle d'affichage de données. Cette boucle permet d'afficher des animations graphiques complexes, réagir des événements synchrones 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).

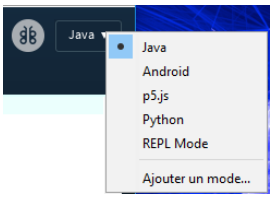
primitives_graphiques

```
/**
 Utiliser des primitives graphiques dans un script Processing

 */

void setup() { // Initialiser le sketch
  size(400,400); // créer une fenêtre de taille 800x600

}

void draw() { // boucle infinie de dessin
  background(0); // redessinier la fenêtre en noir - 0 -> noir, 255 -> blanc
  noStroke(); // ne pas dessiner de contour pour les objets

  // dessiner une ellipse verte
  fill(0,255,0); // vert (Red/Green/Blue)
  ellipse(200,250, 100,100);

  // dessiner un rectangle bleu
  fill(0,0,255);
  rect(50,50,100,150);

  // dessiner un triangle rouge (avec transparence)
  fill(255,0,0,120);
  triangle(20,20, 300,150, 80, 300);
}
```

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 ! Les quelques exercices ci-après vous donnerons un aperçu de ce qui peut être fait en quelques lignes de code.
La structure d'un programme Processing permet aussi d'implémenter **aisément une machine à états** (cf. `https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/Machine_Etats.zip` pour un exemple) !

L'IDE Processing propose plusieurs modes : Java (par défaut) mais aussi Android (ADB doit être installé), p5.js, Python, R et REPL (Read Eval Print Loop 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.



4. exercices (de base)

4.1 des dessins

- Créer une composition graphique en couleur (incluant la transparence) utilisant au moins une ellipse, une ligne et un rectangle
- Dessiner un tableau simplifié de type « *Mondrian* » (père du néo-plasticisme) ou Sophie Taeuber-Arp – voir des exemples ici : **http://www.wikipaintings.org/en/piet-mondrian/composition-a-1923**
  et là : **http://deetrendss.esy.es/tag/sophie-taeuber-arp**

4.2 des images

- Afficher deux images avec une teinte différente
- Charger un fichier **png** avec un bit de transparence et créer une composition graphique en superposant les couches.

4.3 de la typographie

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

5. de l'interaction avec « classe »

5.1 utiliser les entrées « classiques »

- Utiliser les flèches du clavier pour modifier la position d'un triangle affiché dans la fenêtre
- Dessiner deux formes qui réagissent différemment suivant les actions de la souris

5.2 jouer des films vidéo

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

Charger le projet **video**
(`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.

5.3 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()*).

6 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 du journal « *le Monde* » et afficher les « *unes* » dans des cercles quand l'utilisateur clique sur le titre du journal.

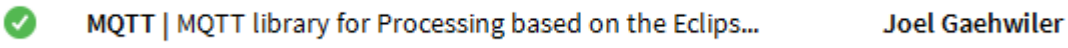
6.2 JSON

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.

6.3 MQTT

Ouvrez Processing, installez la librairie MQTT (Menu « **Outils** » | « **Ajouter un outil** », onglet « **Libraries** » MQTT.



Téléchargez et installez le sketch. Ouvrez maintenant dans un navigateur le site **shiftr.io** à l'adresse **`https://shiftr.io/try`**

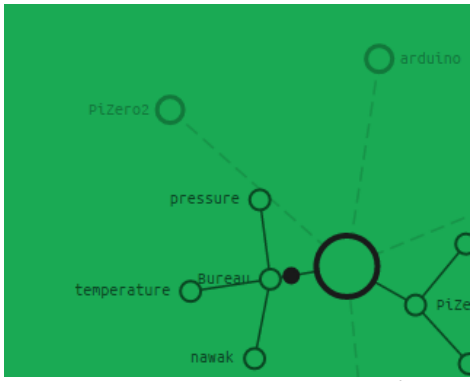


Figure 4 : Architecture MQTT "Publish/Subscribe"

Ecrivez tout d'abord un programme Processing qui génère des valeurs aléatoires de température toutes les secondes et les envoie au broker MQTT (**vous devez déterminer le topic**). Chaque instance lancée sera considérée comme une « pièce » de la maison)
Ecrivez ensuite un programme qui devra s'abonner à l'ensemble des températures émises par chacune des instances et afficher la moyenne générale des températures de la « maison » (générée donc par chacune des pièces) et les moyennes par « pièce ».

7 traitement à partir d'images

7.1 "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`)

7.2 QRCode

A partir de l'exemple téléchargé ici :
`http://www.irit.fr/~Philippe.Truillet/ens/ens/processing/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 : générer un QR Code depuis Processing, voir le code ici :
`https://github.com/truillet/upssitech/blob/master/SRI/1A/Code/QRCode_Generator.zip`

7.3 Réalité mixte - 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)

8. capteurs et multimodalité

8.1 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** (total) de l'écran et le sauvegarder en jpeg.

8.2 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).

8.3 Arduino

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* »)

adresses utiles

- **Processing** : http://www.processing.org
- **P5.js** : http://p5js.org
- **studioSketchpad** : sri-upssitech-p5.sketchpad.cc

- **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
- **Librairies** : https://processing.org/reference/libraries/

- **OpenCV for Processing** : https://github.com/atduskgreg/opencv-processing/releases
- **TTSlib for Processing** : http://www.local-guru.net/blog/pages/ttslib