TP 2 – Processing.org avancé

Ph. Truillet

Janvier 2021 – v. 1.1

Dans ce deuxième volet, nous allons nous intéresser à un certain nombre de concepts et de librairies pour aller plus avant dans la programmation et les modalités d'interaction non-conventionnelles.

1. de l'interaction avec « classe »

1.1 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()*).

1.2 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)

2. la mise en réseau des données

2.1 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.org, 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.**

3. un peu de Réalité Augmentée

3.1 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`

3.2 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)

3.3 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).

3.4 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.org 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`

4. un peu de distribution et de multimodalité

4.1 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.org qui permet de prendre un screenshot de l'écran et le sauvegarder au format jpeg**.

4.2 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 `https://github.com/truillet/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).

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

Utiliser au préalable l'IDE Arduino (**https://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.org.
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* »)

6. 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

- **TTSlib for Processing** : http://www.local-guru.net/blog/pages/ttslib