processing, shell, code, em

Traitement par lot avec Processing en ligne de commande

Le script ci-dessous permet de traiter un répertoire d'images et d'enregistrer le résultat dans un second répertoire.
Vu qu'il s'agit de traitement par lot, on peut aussi le démarrer en ligne de commande, en mode “headless”, c'est à dire sans ouvrir de fenêtre. Une fois le script terminé, il s'arrêtera automatiquement.

Pour utiliser le mode headless, sur linux il faut installer le paquet **xvfb** (X Virtual FrameBuffer et ses dépendances. Il faut aussi avoir java d'installé

```
# Debian 9.5 @ Kirin
sudo apt-get install xvfb libxrender1 libxtst6 libxi6
sudo apt-get install default-jdk # ou sudo apt-get install default-jre
```

Ensuite on peut démarrer le script de cette manière en passant en argument le dossier d'origine et le dossier de destination :

```
 xvfb-run /home/emoc/processing-3.5.3/processing-java --sketch="/home/emoc/sketchbook/2020_KI/traitement_image_par_lot_001/" --run
"/home/user/dossier_orig" "/home/user/dossier_dest"
```

Code

C'est une base qui fonctionne mais beaucoup d'améliorations pourraient être apportées!

traitement_image_par_lot.pde (cliquer pour afficher le code)

traitement_image_par_lot.pde

```
/*
  Base pour faire du traitement d'images par lot
  processing 3.5.3 @ kirin / Debian Stretch 9.5
  20200508 / pierre@lesporteslogiques.net

  charger toutes les images d'un dossier une par une
  leur appliquer une transformation
  les enregistrer
  (le dossier ne doit contenir que des images que processing peut lire : jpg, png, tif, tga)

  utilisable en ligne de commande :
    xvfb-run /home/emoc/processing-3.5.3/processing-java --sketch="/home/emoc/sketchbook/2020_KI/traitement_image_par_lot_001/" --
run "/home/user/dossier_orig" "/home/user/dossier_dest"
*/

boolean GUIMODE = true;       // GUI ou ligne de commande ? Changé automatiquement si le script est lancé en ligne de commande

String dossier_orig = "/home/emoc/sketchbook/2020_KI/test"; // dossier des images à traiter
String dossier_dest = "/home/emoc/sketchbook/2020_KI/res";  // dossier des images transformées
String[] fichiers_a_traiter;  // liste des fichiers du répertoire à traiter
int nb_fichiers = 0;          // nombre de fichiers à traiter

PImage img_orig;              // image à traiter
PGraphics img_dest;           // image résultant du traitement

String fichier_orig = "";     // nom du fichier original à traiter
String fichier_dest = "";     // nom du fichier à créer
String chemin_orig = "";      // chemin complet vers le fichier original
String chemin_dest = "";      // chemin complet vers le fichier à créer
String extension = "png";     // extension et format de fichier à créer
String racine = "image";      // racine du nom de fichier à créer

int compteur = 1;             // numéro du premier fichier, les autres fichiers seront nommés à partir de là
String numero;                // formatage du nombre contenu dans le nom de fichier à créer

void setup() {
  size(800, 300);
  init();                     // traitement des arguments associés à la ligne de commande
}
```

```
void draw() {

  println("dossier à traiter : " + dossier_orig);
  println("dossier des fichiers traités : " + dossier_dest);
  fichiers_a_traiter = listFileNames(dossier_orig);
  printArray(fichiers_a_traiter);
  nb_fichiers = fichiers_a_traiter.length;

  for (int i = 0; i < nb_fichiers; i++) {

    chemin_orig = dossier_orig + "/" + fichiers_a_traiter[i];
    numero = nf(compteur+i, 4); // numero du fichier formaté 0001, 0002, etc.
    fichier_dest = racine + "_" + numero + "." + extension;
    chemin_dest = dossier_dest + "/" + fichier_dest;
    println("traitement du fichier " + chemin_orig);
    println("fichier à créer : " + chemin_dest);

    img_orig = loadImage(chemin_orig);

    img_dest = createGraphics(img_orig.width, img_orig.height);
    img_dest.beginDraw();
    img_dest.image(img_orig, 0, 0);
    img_dest.noStroke();
    img_dest.fill(255, 0, 0);
    img_dest.ellipse(img_dest.width/2, img_dest.height / 2, img_dest.width / 2, img_dest.height / 2);
    img_dest.endDraw();
    img_dest.save(chemin_dest);
  }

  if (!GUIMODE) {
    exit();
  }
  noLoop();
}


// Fonction pour traiter les arguments de la ligne de commande
void init() {
  if (args != null) {
    GUIMODE = false;
    println("Arguments : " + args.length);
    for (int i = 0; i < args.length; i++) {
      println(args[i]);
    }
    if (args.length == 2) {
      dossier_orig = args[0];
      dossier_dest = args[1];
    } else {
      println("arguments insuffisants (indiquer dossier de départ et dossier d'arrivée)");
      exit();
    }
  } else {
    println("pas d'arguments transmis par la ligne de commande");
  }
}

// fonction pour lister les fichiers d'un dossier, renvoie un tableau de chaines de caractères
// d'après Daniel Shiffman : https://processing.org/examples/directorylist.html
String[] listFileNames(String dir) {
  File file = new File(dir);
  if (file.isDirectory()) {        // C'est un dossier
    String names[] = file.list();
    return names;
  } else {                         // If it's not a directory
    return null;
  }
}
```

Ressources

- https://github.com/processing/processing/wiki/Running-without-a-Display
- https://forum.processing.org/two/discussion/23924/handling-command-line-arguments-in-a-sketch

Article extrait de : http://lesporteslogiques.net/wiki/ - **WIKI Les Portes Logiques**
Adresse : **http://lesporteslogiques.net/wiki/ressource/code/processing/traitement_par_lot?rev=1589054094**
Article mis à jour: **2020/05/09 21:54**