CNAM NSY116 - Multimédia et interaction humain-machine
ED 8 - Tag clouds

Exercice 1 : Fonctions de base pour la typographie avec Processing

Ecrire un programme produisant l'image ci-dessous (à gauche). En particulier, on cherche comment obtenir la boîte englobante d'un texte écrit dans une police quelconque.

Exercice 2 : Occurences de mots dans un texte

Les nuages de mots (tag clouds) sont généralement produits en ligne sur des sites web participatifs, par les utilisateurs de ces sites. Les mots descripteurs employés sont laissé libres et ne correspondent que rarement à des thésaurus professionnels. On peut aussi employer cette technique de visualisation pour des mots "représentatifs" du texte (ou d'un corpus de textes) d'un même auteur. Par exemple, ne retenir que les noms propres ou communs les plus féquents.

Un premier nettoyage du fichier texte peut être fabriqué par un utilitaire d'expression régulière. Comment ?

On pourra appliquer la technique à des textes extraits de la bibliothèque numérique ABU en ligne au CNAM (http://abu.cnam.fr), par exemple avec le "Discours de la méthode" de Descartes.

Exercice 3 : Affichage du nuage de mots

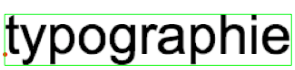
On affecte à chaque mot de la liste obtenue dans l'ex. 2, une taille sur l'écran proportionnelle à la fréquence du mot. Reste à déterminer sa position à l'écran. On propose ici de partir d'une position au hasard. Il faudra ensuite tenter de résoudre les cas de chevauchement.

1) Quelle est la condition pour que 2 boîtes englobantes se chevauchent ?

2) Que faire pour (le plus simplement possible) gérer les chevauchements ?

3) Ecrire le programme final. Explorer son comportement en fonction de la taille de l'image et du nombre de tags.

4) Proposer des variantes



Exercice 1 :

```
size(600,300);
smooth();
textAlign(LEFT);
PFont font = loadFont("ArialMT-48.vlw");
// creer ce fichier avec le menu Tools/create font

background(255);
stroke(0);fill(0);

int x = 100;
int y = 150;
String lemot = "typographie";

textFont(font,60);text(lemot,x,y);

stroke(255,0,0);strokeWeight(5);point(x,y);

// mauvaise solution
noFill();strokeWeight(1);stroke(255,0,0);
rect(x,y-60,textWidth(lemot),60);

// bonne solution
noFill();strokeWeight(1);stroke(0,255,0);
float h = textAscent()+textDescent();
rect(x,y-textAscent(),textWidth(lemot),h);

save("exo1.png");
```

Exercice 2 :

exemple de script awk, utilisé après aspiration d'une page HTML quelconque

```
{
      # suppression des balises html
      gsub(/<.*>/,"")
      # suppression des caracteres indesirables
      gsub(/[.,:;!?(){}_«»\[\]]/, "")
      # gestion de l'apsotrophe
      gsub(/'/, "e ")
      # mise en minuscule
      tolower($0)
      # maj freq des mots du parag courant
      for (i = 1; i <= NF; i++) {
        count[$i]++
      }
    }

END {
        for (w in count) {
        print count[w],w | "sort -nr"
        }
    }
```

ou sinon, en utilisant par exemple le fichier http://abu.cnam.fr/cgi-bin/freq?methode3 et en nettoyant les mots inutiles, on obtient :

```
46 coeur
46 cause
39 vérité
38 corps
36 sang
```

```
35 nature
34 Dieu
31 hommes
28 monde
27 opinions
23 temps
23 pensées
20 dessein
20 connoissance
19 vie
17 terre
17 sciences
17 esprits
16 matière
16 coutume
15 principes
26 esprit
13 public
13 méthode
13 artères
13 actions
12 veines
12 philosophie
12 parties
12 chaleur
11 lois
11 expériences
11 animaux
10 organes
10 occasion
10 lumière
10 âme
10 homme
10 fondements
10 démonstrations
```

Exercice 3 :

```
PFont font;
String fontFile = "ArialMT-72.vlw";
int maxfonte = 60;
int minfonte = 10;
Tag[] tags = new Tag[100];
int nTags;

void setup(){
  size(600,400);
  smooth();
  textAlign(LEFT);
  font = loadFont(fontFile);
  // lecture du fichiers des tags
  String lines[] = loadStrings("methode3.txt");
  for (int i=0; i < lines.length; i++) {
   String params[] = split(lines[i]," ");
   tags[i] = new Tag(params[1],int(params[0]));
  }
  nTags = lines.length;
  // calcul dimensions et position initiale des tags
  int maxpoids = 0; int minpoids = 0;
  for (int i=0; i<nTags; i++) {
   maxpoids = int(max(maxpoids,tags[i].poids));
   minpoids = int(min(minpoids,tags[i].poids));
  }
  for (int i = 0; i < nTags; i++) {
    tags[i].tf = int(map(tags[i].poids, minpoids, maxpoids, minfonte, maxfonte));
    textFont(font, tags[i].tf);
    tags[i].h =  int(textAscent()+textDescent());
```

```
    tags[i].w = int(textWidth(tags[i].mot));
    tags[i].bouge();
  }
}

void draw(){
  // recherche des collisions
  int chocs = 0;
  for(int i=0; i<nTags; i++){
   for (int j=i+1; j<nTags; j++) {
     if (tags[i].mecoupe(tags[j])) {
       tags[j].bouge();
       chocs++;
       println("choc "+chocs+" moi "+i+" lui "+j);
     }
   }
  }
  // dessins des tags
  background(255);
  for(int i=0; i<nTags; i++){
    tags[i].dessin();
  }
  //test d'arret
  println("FINI "+frameCount);
  if (chocs == 0) {saveFrame("resu####.png");noLoop();} else chocs = 0;
}



class Tag{
  int x, y, w, h, poids, tf; String mot;

  Tag(String lemot, int lepoids){
    mot = lemot;poids = lepoids;
    x = y = w = h = tf = 0;
  }

  void bouge(){
   x = int(random(0,width-w));
   y = int(random(h,height));
  }

  void dessin(){
    fill(0);textFont(font,tf);text(mot,x,y);
    //noFill();stroke(0);rect(x,y-h,w,h);
   }

  boolean mecoupe(Tag l){
   return ! ( l.x > x+w || l.x+l.w < x || l.y-l.h > y || l.y < y-h );
  }
}
```