TP 11 Boucles, animations, etc

Informations techniques PC Suse :

(a) Pour démarrer une session : utilisateur **licencep** et mot de passe **7002n***.

(b) Pour démarrer *Processing* : clic sur `l'icône lézard` en haut à droit → Dévelopemment → Processing.

(c) Pour démarrer un *terminal* : `l'icône lézard` → Terminal → Konsole.

(d) Pour ouvrir un gestionaire/navigateur de fichiers : `l'icône lézard` → Système → Dolphin

(e) Pour modifier un fichier, clic droit sur le fichier → Ouvrir avec Kate

1 Commandes Linux et un petit programme Java

Démarrer une console/terminal en suivant les instructions au point (c) ci-dessus.
Démarrer un gestionnaire de fichiers en suivant les instructions au point (d) ci-dessus.
Vous allez vous servir à la fois du terminal et du gestionnaire graphique de fichiers.

Exercice 1 Taper dans le terminal :

1. une commande `cd` (un acronyme pour l'anglais *change directory*) pour se placer dans le dossier `/tmp/`.
2. Par la suite, taper une commande `mkdir` (*make diretory*) pour créer un dossier à l'intérieur du dossier `/tmp/`
3. Finir avec une commande `cd` pour se placer dans le dossier que vous venez de créer.

N'hésitez pas à regarder les exemples à la fin du Cours 3-4 sur le systèmes, disponibles à `cedric.cnam.fr/~porumbed/vari1`

Exercice 2 Taper la commande ci-après pour télécharger le fichier source `Toto4.java`. Compiler et exécuter ce programme à l'aide des indications en haut à droite.

```
wget  cedric.cnam.fr/~porumbed/Toto4.java
```

Pour **compiler** le programme Java, taper :

```
javac   Toto4.java
```

Et pour l'**exécuter :**

```
java   Toto4
```

Exercice 3 Aller dans le gestionnaire de fichiers et chercher le dossier que vous avez crée à l'exercice 1. Cliquer sur ce dossier pour trouver `Toto4.java` à l'intérieur. Clic droit sur `Toto4.java` et choisir `Ouvrir avec` → `kate`.

- Regarder le programme dans l'éditeur `kate`, identifier les instructions d'affichage.
- Ajouter 4-5 lignes pour faire le programme calculer la valeur minimale.

Exercice 4 Utiliser la commande `mv` pour changer le nom du dossier crée au premier exercice. Attention : il faut se placer dans `/tmp/` (voir commande `cd ..`) pour pouvoir lancer `mv`.

2 Boucles et animations

Exercice 1 Écrire un programme qui affiche 19 cercles de tailles aléatoires, placés aléatoirement. Chaque cercle doit être rempli avec une autre nuance de rouge, c.-à.d. n'hésitez pas à utiliser `fill(random(...),0,0)`.

Exercice 2 On considère deux tableaux `x` et `y`, pre-remlis avec des valeurs, comme dans le programme à droite. Continuer le programme et écrire une boucle `for` qui déplace la fenêtre aux positions (x[0],y[0]), (x[1],y[1]), (x[2],y[2]), ..., avec une pause d'une demi-seconde après chaque déplacement.

Exercice 3 Ajouter une boucle extérieure `while(true)` qui sert à faire tourner en boucle les déplacements de fenêtre de l'exercice précédent.

```
void setup(){
    int[] x = {200,373,373,200,27, 27};
    int[] y = {0,  100,300,400,300,100};
    int i = 0;
    surface.setLocation(x[i],y[i]);
    delay(500); //pause 0.5 secondes
    //boucle for à ajouter
}
```

Exercice 4 À partir de cet exercice, on n'utilise plus de tableau. Écrire un programme avec une boucle `for` qui déplace la fenêtre successivement aux coordonnées (x, y) =(0,0), (0,10), (0,20), ... Après chaque mouvement

de fenêtre, il faut faire une pause (delay) de 50 milli-secondes. Arrêter le mouvement lorsque la coordonnée verticale y arrive à la valeur `displayHeight` (hauteur d'écran) moins la hauteur de la fenêtre.

Exercice 5 Continuer l'exercice précédent pour changer le sens de déplacement de la fenêtre après avoir arrivé au bord en bas. Autrement dit, après avoir descendu à la plus basse position, la fenêtre doit commencer à monter, avec la même vitesse (de 10 en 10 avec une pause de 50 milli-secondes à chaque fois). Elle doit changer encore une fois de direction lorsqu'elle arrive de nouveau aux coordonnées (0,0). Ajouter une boucle extérieure `while(true)` qui permet d'exécuter en boucle l'aller-retour (du haut en bas et retour) de la fenêtre.

Exercice 6 Écrire un programme qui affiche une fenêtre de taille 700×700 avec une grille. Une cellule de la grille doit avoir la taille 70×70. INDICATION : afficher 10 lignes avec 10 rectangles par ligne. N'hésitez pas à consulter les programmes présentés au cours 11, disponibles en ligne à `cedric.cnam.fr/~porumbed/vari1/`.

Exercice 7 On veut réaliser une animation similaire à celle du programme `balle.pde` présenté dans le cadre du cours 10, disponible à `cedric.cnam.fr/~porumbed/vari1/progsc10.zip`. Implémenter 4 étapes :

1. déclarer une variable globale a
2. Écrire une méthode `void setup()` qui initialise $a = 0$ et génère une fenêtre/surface de taille 900×900 à l'aide de `size(...)`. Initialiser un `frameRate` de 5 affichages par seconde.
3. Écrire une méthode `void draw()` qui permet d'afficher un cercle de rayon 25 aux coordonnées (a, a) où a est la variable globale déclarée au point 1 ci-dessus et initialisée au point 2.
4. Modifier la méthode `draw()` ainsi :
   — au début du `draw()`, faire effacer la surface de dessin avec l'instruction `background(204,204,204)` ;
   — faire incrémenter a à chaque itération à la fin du `draw()`.
5. Modifier le programme pour faire la balle rebondir dans le coin en bas à droite, c.à.d, la balle doit d'abord toucher le coin en bas à droite et par la suite repartir vers le coin en haut à gauche.

Exercice 8 Télécharger l'image `balle.jpeg` à l'aide de la commande
`wget cedric.cnam.fr/~porumbed/balle.jpeg`

Sauvegarder le programme précédent et donner lui le nom `toto`. Ainsi, `Processing` va créer un dossier appelé `toto`, probablement dans le dossier `sketchbook` du dossier personnel. Mettre l'image `balle.jpeg` dans ce dossier `toto`. Inspirez vous du code ci-après pour afficher la balle, voir aussi les derniers programmes du cours 11.

```
void setup(){
    PImage img = loadImage("balle.jpeg");
    image(img,0,0);
}
```

Pour finir, remplacer le cercle du programme précédent avec la balle ! L'objectif est d'obtenir une balle qui se déplace le long de la diagonale, en faisant des aller-retours le long de la diagonale.

Exercice 9 Exécuter le programme ci après. Observer que la fonction `toString(int[]..)` permet de transformer un tableau en chaine de caractères, pour qu'on puisse l'afficher facilement. C'est une fonction déjà écrite dans la bibliothèque (librairie) standard `java.util.Arrays`. Écrire une fonction `tabString(int[] t)` qui permet de réaliser la même transformation. Tester si `tabString(...)` mène exactement au même affichage que `java.util.Arrays.toString(...)`, c.-à.-d, les deux doivent faire apparaître les crochets.

```
void setup(){
    int[] tab = {9,12,1,7,2,11, -3, 14, 18};
    String tabPourAffichage = java.util.Arrays.toString(tab);
    println(tabPourAffichage);
}
```