TP 10 Boucles, animations, casse-briques

Informations techniques PC Suse :

(a) Pour démarrer une session : utilisateur **licencep** et mot de passe **7002n***. Vous trouverez :
— un gestionnaire de fichiers en haut à gauche placé dans le dossier personnel HOME
— une `icône lézard` en haut à droite pour accéder au menu.

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

(c) Pour démarrer une *console* : clic sur `l'icône lézard` en haut à droit → Terminal → Konsole.

(d) Pour ouvrir un gestionaire/navigateur de fichiers : clic sur `l'icône lézard` → Utilitaires → Dolphin.

(e) Pour lancer une commande : Alt + Space

(f) Pour modifier un fichier, clic droit sur le fichier → Ouvrir avec → Kate (ou autre éditeur de votre choix).

1 Commandes Linux et un petit programme Java

Exercice 1 Ouvrir un terminal, taper les commandes ci-après et indiquer le résultat de chaque commande.

```
ls /bin/z*
ls /bin/[y-z]*
ls /bin/*t

echo -e "aaa\nbbb\nccc" > fic.txt

grep a fic.txt
```

Exercice 2 Taper une commande `mkdir` (*make diretory*) pour créer un dossier appelé `tp10`. Taper 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 2 sur le systèmes, disponibles à `cedric.cnam.fr/~porumbed/vari1`

Exercice 3 Taper la commande ci-dessous pour télécharger le fichier source `Toto5.java`.
`wget cedric.cnam.fr/~porumbed/Toto5.java`

Pour **compiler** le programme Java, taper :
`javac Toto5.java`
Et pour l'**exécuter :**
`java Toto5`

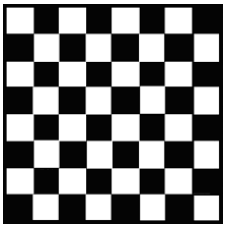
Exercice 4 Aller dans le gestionnaire de fichiers et chercher le dossier que vous avez crée à l'exercice 1. Cliquer sur ce dossier pour trouver `Toto5.java` à l'intérieur. Clic droit sur `Toto5.java` et choisir `Ouvrir avec` → `kate`.

— Regarder le programme dans l'éditeur `kate`, identifier l'instruction qui affiche le tableau.
— Ajouter une boucle `for` pour faire le programme calculer et afficher la valeur minimale du tableau.

2 Boucles, fonctions, casses-briques

Exercice 1 Écrire un programme `Processing` qui trace en continu dans une zone de 600×600 pixels un cercle non remplis de diamètre 30 pixels, centré à la position de la souris.
Rappel : utiliser *mouseX* et *mouseY* pour récupérer la position de la souris.
Attention : n'oubliez pas d'effacer la toile via `background`(...) avant d'afficher le cercle.

Exercice 2 Écrire une fonction `afficherMatSurLaToile(int[][] m)` qui permet d'afficher les valeurs d'une matrice m de taille 3×3 au milieu d'une toile de taille 500×500. On doit obtenir 3 lignes et 3 colonnes.
N'hésitez pas à regarder les exemples de la section sur les matrices du Cours 10, disponible en ligne à : `cedric.cnam.fr/~porumbed/vari1`



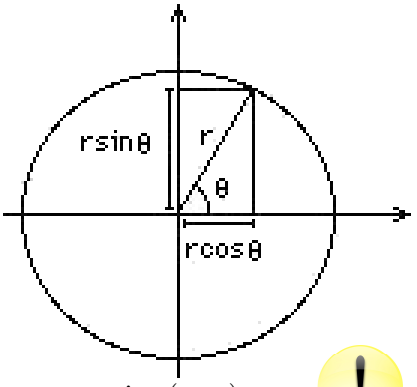
Exercice 3 Écrire un programme pour dessiner la table d'échecs à droite. Vous allez utiliser deux boucles `for` imbriquées pour parcourir les cases de la table. Pour une case noire on doit utiliser `fill(0,0,0)` et pour une case blanche `fill(255,255,255)`. Une case (i, j) est blanche lorsque la condition suivante est satisfaite $(i + j)\%2 == 0$.

Exercice 4 Modifier le programme précédent pour obtenir l'effet suivant. Lorsque l'utilisateur survole la table avec la souris, on affiche un cercle qui couvre (cache) la table. Lorsque la souris quitte la table, elle est de nouveau affichée (c. à. d., le cercle disparait).

Exercice 5 Commencer à écrire la fonction `setup()` et mettre une fenêtre de taille 120×120. Ajouter une boucle `for` pour déplacer la fenêtre sur *une trajectoire circulaire* à l'aide de `surface.SetLocation((int)x,(int)y)`. La fenêtre doit suivre une trajectoire décrite par les coordonnées :

$$x = centreX + r \cdot sin(\theta) \text{ et } y = centreY + r \cdot cos(\theta),$$

où θ représente un angle en radians. En fait, θ est incrémenté graduellement de 0 à 2π (rappel : 2π représente un angle de 360 degrés), voir aussi la figure ci-contre.



Par exemple, la trajectoire pourrait être :

$(300 + 100 \cdot sin(0 \cdot \frac{\pi}{12}), 300 + 100 \cdot cos(0 \cdot \frac{\pi}{12}))$,
$(300 + 100 \cdot sin(1 \cdot \frac{\pi}{12}), 300 + 100 \cdot cos(1 \cdot \frac{\pi}{12}))$,
$(300 + 100 \cdot sin(2 \cdot \frac{\pi}{12}), 300 + 100 \cdot cos(2 \cdot \frac{\pi}{12}))$,
...
$(300 + 100 \cdot sin(12 \cdot \frac{\pi}{12}), 300 + 100 \cdot cos(12 \cdot \frac{\pi}{12}))$.

Attention : les coordonnées non entières comme `100*sin(...)`) doivent être converties (cast) en valeurs entières en ajoutant `(int)` devant la formule ; ex. : `int x = (int) (300+100*sin(1))`.

Exercice 6 Soit le code ci-après.

1. Écrire la fonction `trierTab` pour la faire trier le tableau `tab` en **ordre décroissant** en utilisant la méthode présentée au cours 10, voir site web.
2. Continuer la fonction `setup()` pour ajouter à la fin le code nécessaire pour afficher le tableau trié dans la console, **ainsi que sur la toile de dessin**, au mileu .

```
int[] tab;
void trierTab(){
    //à remplir
}
void setup(){
    size(800,600);
    tab = new int[20];
    for(int i=0;i<tab.length;i++)
        tab[i] = 15+(int)random(50);
    trierTab();
    \\à remplir
}
```

Exercice 7 Écrire une fonction

```
float racine(float x)
```

qui calcule et renvoie la raciné carré de x. Utiliser la suite convergente suivante , connue depuis l'antiquité (Héron d'Alexandrie) :

$$r_{n+1} = \frac{r_n + \frac{x}{r_n}}{2}$$

La première valeur de la série est $r_1 = x$. Vous n'avez pas besoin d'utiliser un tableau, mais juste d'exprimer la nouvelle valeur r_{n+1} en fonction de la valeur précédente r_n. Utiliser une boucle `for` pour calculer la 5ème valeur de cette série, qui devrait être pas trop éloignée de la vraie racine carré. Appeler `racine(200)` dans la fonction `setup` et comparer le résultat avec `sqrt(200)`.

Exercice 8 Écrire une fonction

```
float racine(float x, float erreurMax)
```

qui calcule la série indiquée plus haut jusqu'à ce que $r_n - r_{n+1} < erreurMax$. Il est conseillé d'utiliser une boucle comme `while(rAncien-rNouveau<erreurMax)....`

Exercice 9 Aller sur le site web de l'UE Vari1 `cedric.cnam.fr/~porumbed/vari1` et télécharger les programmes présentés dans le cadre du cours 10. Extraire le fichier `casseBriques.pde` et :

1. Copier le code dans la fenêtre de `Processing`
2. Sauvegarder le fichier avec le nom `casseBriquesVersion2.pde`.
3. Copier le fichier `balle.jpeg` dans le dossier `casseBriquesVersion2`
4. Exécuter ce programme. Remarquer qu'on a une seule rangée de briques à détruire.

Modifier le programme pour le faire afficher/tracer **3 rangées de briques**. Le programme devrait afficher `Gagné` uniquement si toutes les briques ont été cassées.

Indication : remplacer le tableau `briques` par une matrice de 3 lignes et 8 colonnes.