TP 12 Boucles, animations, casse-briques

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

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 directory*) 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 3-4 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`

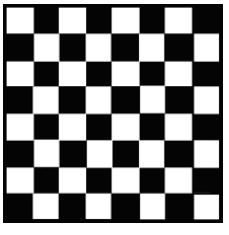
Exercice 4 Aller dans le gestionnaire de fichiers et chercher le dossier que vous avez crée à l'exercice 1. Cliquer sur ce dossier pour trouver `Toto5.java` à l'intérieur. Clic droit sur `Toto5.java` et choisir `Ouvrir avec` → `kate`.

- 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 méthode `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 12, 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 $(i + j)\%2 == 0$.

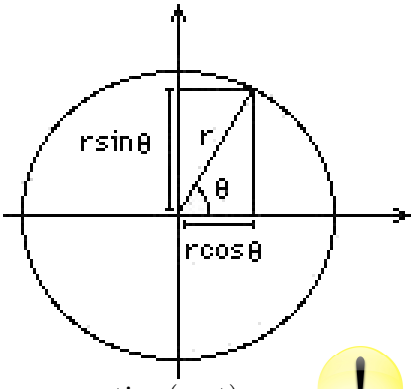
Exercice 4 Modifier le programme précédent pour obtenir l'effet suivant. Lorsque l'utilisateur survole la table avec la souris, on affiche un cercle qui couvre (cache) la table.

Exercice 5 Commencer à écrire la méthode `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 :

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



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 Écrire une fonction qui renvoie la moyenne de trois arguments de type `int`.

Exercice 7 Modifier la fonction précédente pour calculer la moyenne de trois nombres, mais en ignorant du calcul les valeurs nulles. Par exemple, si on passe les valeurs 3, 0, 5, alors la fonction doit renvoyer 4.

Exercice 8 La fonction ci-après permet de calculer le produit $1 \times 2 \times 3 \cdots \times n$, où n et l'argument passé à la fonction.

```
int calcFactoriel(int n){
    int prod = 1;
    for(int i=2;i<=n;i++){
        prod = prod * i;
    }
    return prod;
}
```

Écrire une fonction `somme` qui reçoit un argument `n` et qui calcule et renvoie la valeur de la somme $1 + 2 + 3 + 4 + \cdots + n$.

Exercice 9 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'écrire 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 proche de la vraie racine carré. Appeler `racine(200)` dans la méthode `setup` et comparer le résultat avec `sqrt(200)`.

Exercice 10 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 11 Aller sur le site web de l'UE Vari1 `cedric.cnam.fr/~porumbed/vari1` et télécharger les programmes présentés dans le cadre du cours 12. 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.

A) Modifier le programme pour utiliser des briques de longueur 200 et une palette de longueur 150.
B) Modifier le programme pour détruire les briques en 2 étapes. Lorsque la balle touche une brique une première fois, la brique devient jaune sans disparaître. Lorsque la balle touche la brique une deuxième fois, la brique disparaît (est cassée).
C) **BONUS** Modifier le programme pour ajouter trois couches de briques.