Techniques des Logiciels Interactifs

durée indicative : **3h**
documents autorisés : **1 feuille A4 recto-verso manuscrite de votre main**

consignes :
Le barème est **indicatif**. La qualité de la **présentation**, de l'**expression**, de l'**orthographe** sera prise en compte dans la notation de **manière significative**.
Si le sujet présente des ambiguïtés, précisez vos choix. Il sera tenu compte de vos hypothèses. Lorsqu'il est demandé de produire du code, ne vous focalisez pas sur la correction de la syntaxe.

Généralités (10 points)

Question 1
a) Définissez "interface" et "interaction".
b) Expliquez la différence entre ces deux termes dans le contexte de l'Interaction Homme-Machine (IHM).

Question 2
a) Comparez l'évolution dans le temps des performances des ordinateurs et celles des humains.
b) Situez dans ce cadre la problématique de l'IHM.

Question 3
Situez dans le temps (donnez la décennie de) :
• l'invention de la souris ;
• l'invention des interfaces graphiques utilisant la métaphore du bureau.

Question 4
Donnez trois styles d'interaction et pour chacun d'entre-eux un exemple de système l'utilisant.

Question 5
a) Pourquoi faut-il séparer le noyau fonctionnel de l'interface lors de la réalisation d'une application ?
b) Quel mécanisme permet au noyau fonctionnel d'appeler des fonctions de l'interface tout en le maintenant indépendant de cette dernière ?

Programmation par événements (6 points)

Question 6

Vous devez réaliser l'interface graphique d'un logiciel qui permet de manipuler des photos sur une surface interactive en utilisant les doigts comme pointeurs.
Ceux-ci génèrent trois types d'événements : ↓ lorsqu'un doigt entre en contact avec la surface ; M lorsqu'il se déplace au contact de la surface ; et enfin ↑ lorsqu'il quitte cette dernière.
Ces événements contiennent la position (`Point2D e.getPoint()`) où ils ont eu lieu ainsi qu'un identifiant du doigt utilisé (`int e.getFingerId()`).
On veut réaliser l'interaction suivante :

- avec 1 doigt, la photo est déplacée, le déplacement est géré par les fonctions données suivantes : `start_drag(Point2D start)` qui définit le point de départ du déplacement, `drag(double dx, double dy)` qui déplace la photo de `dx` (resp. `dy`) pixels horizontalement (resp. verticalement), et `end_drag()` qui termine l'interaction ;

- avec 2 doigts, la photo est déplacée/tournée/zoomée, le redimensionnement est géré par la fonction donnée suivante : `pinch(Point2D old_p1, Point2D p1, Point2D old_p2, Point2D p2)` où old_p1 (resp. p1) désigne la position précédente (resp. courante) de l'un des doigt et old_p2 (resp. p2) celle de l'autre doigt ;
- si on relâche un des deux doigts, on revient en mode déplacement.

Dessinez la machine à état qui réalise ces interactions (vous pouvez faire l'hypothèse qu'au plus 2 doigts seront utilisés en même temps).
Notez que les évènements arrivent sérialisés : si deux doigts arrivent en même temps, une séquence de deux ↓ successifs (avec des identifiants différents) est produite ; de même, si deux doigts bougent en même temps, deux M successifs seront produits.

Question 7

Réalisez en java la machine à état de la question précédente.
Pour cela, vous est donnée la classe `Surface` qui utilisera votre classe `Interaction` de la manière suivante :

```
package exam.ui;

import javafx.scene.canvas.Canvas;
import javafx.geometry.Point2D;

public class Surface extends Canvas {
   public Surface() {
      addEventHandler(MouseEvent.ANY, new Interaction(this));
      ...
   }

   public void start_drag(Point2D start) { ... }
   public void drag(double dx, double dy) { ... }
   public void end_drag() { ... }

   public void pinch(Point2D old_p1, Point2D p1,
                     Point2D old_p2, Point2D p2) { ... }
   ...
}
```

Votre classe `Interaction` pourra commencer ainsi :

```
package exam.ui;

public class Interaction ... {
      protected Surface surface;
      public Interaction(Surface surface) {
            this.surface = surface;
            ...
      }
      ...
}
```

Les doigts génèrent en effet des évènements souris (`MouseEvent`) qui ont simplement une méthode supplémentaire pour connaître l'identifiant du doigt correspondant.
Les `MouseEvent e` ont ainsi les méthodes utiles suivantes :

- `Point2D e.getPoint()` qui retourne le lieu de l'événement ;
- `double e.getX()` et `double e.getY()` qui retournent la même information ;
- `int e.getFingerId()` qui retourne un entier qui sera le même pour tout événement entre un ↓ et un ↑ (inclus) si et seulement si c'est le même doigt qui l'a provoqué.
- `String e.getEventType().getName()` qui retourne le type d'évènement dans une chaîne de caractères : `"TOUCH_STARTED"` pour ↓, `"TOUCH_ENDED"` pour un ↑ et `"TOUCH_MOVED"` pour un M.

L'interface fournie par `javafx.event` permettant de recevoir les événements est donnée ci-dessous :

```
public interface EventHandler<T> {
      public void handle(T e);
}
```

Composants interactifs (4 points)

Question 8

Les figures ci-dessous donnent la fenêtre de l'application Mail sous Mac OS X avant et après un redimensionnement. Sachant que la largeur de la colonne de gauche et que la hauteur de la liste des messages peuvent être modifiées par l'utilisateur, donnez l'arbres des widgets que vous utiliseriez pour coder cette interface avec JavaFX.
Certains aspects ne peuvent être rendus simplement avec JavaFX : les boutons collés dans la barre de boutons pourront être considérés comme n'étant pas collés.
Ne détaillez pas le contenu des listes de messages et de boîtes à lettre, ni l'affichage du message lui-même.