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 ?

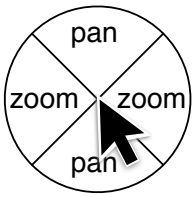
Programmation par événements (6 points)

Question 6
Vous allez spécifier et réaliser le comportement d'une interaction, le ***control-menu***, qui permet de (dé)zoomer et de déplacer un graphique. Celui-ci fonctionne de la manière suivante :

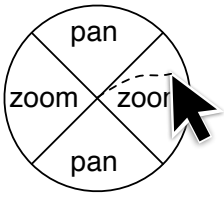
- lorsque l'utilisateur enfonce le bouton de la souris (évènement noté ↓), un menu circulaire divisé en quatre secteurs apparait sous le curseur de la souris ;
- lorsque la souris bouge, avec le bouton toujours enfoncé, (évènement noté M), il ne se passe rien tant que le curseur de la souris ne sort pas du cercle ;
- lorsque le curseur sort du cercle, le menu disparaît et une interaction contrôlée par les déplacements du curseur commence : un (dé)zoom si il sort par les côtés droit ou gauche ou un déplacement si il sort par les côtés haut ou bas.
- l'interaction se prolonge jusqu'au relâché du bouton de la souris (évènement noté ↑).

Soit, en images :

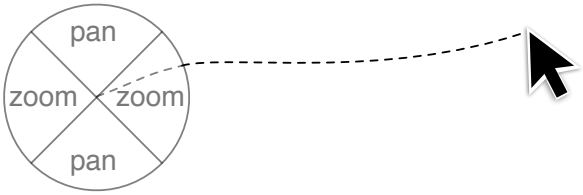
- le menu au départ



- le menu pendant la sélection de l'interaction



- le menu qui a disparu et l'interaction de zoom contrôlée par le déplacement du curseur



Le déplacement est géré par les fonctions suivantes, fournies :
- `void begin_pan(Point2D start)` avec `start` le point initial (centre du menu) au début du déplacement ;
- `void pan(double dx, double dy)` avec `dx` et `dy` les déplacements horizontaux et verticaux du curseur au cours du déplacement ; et
- `void end_pan()` pour la fin du déplacement.

Le zoom est géré par les fonctions suivantes (qui prennent les mêmes paramètres) fournies :
- `void begin_zoom(Point2D start)` au début du déplacement ;
- `void zoom(double dx, double dy)` au cours du déplacement ; et
- `void end_zoom()` pour la fin du déplacement.

Le rayon du cercle est de 40 pixels, pour déterminer par quel côté le curseur sort, on peut remarquer qu'il sort par la droite ou la gauche si et seulement si son déplacement depuis le centre du menu est plus important horizontalement que verticalement (en valeurs absolues).

a) **Dessinez la machine à états qui spécifie cette interaction.**

On rappelle que les évènements ont une méthode `getPoint()` qui retourne un `Point2D` avec la position du curseur. Ce point à des méthodes `getX()` et `getY()` qui en donnent les coordonnées, et une méthode `distance(Point2D p)` qui donne sa distance à un autre

point. On dispose également de la fonction `abs(double i)` qui retourne la valeur absolue d'un entier.
Il n'est pas demandé de gérer le feedback (affichage et masquage du menu).

b) Réalisez en java cette machine à états.

Pour cela, vous est donnée une classe `Canvas` qui utilisera votre classe `Interaction`. C'est cette dernière que vous devez réaliser. Elle pourra commencer ainsi :

```
package exam.ui;

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

Le code, fourni, qui l'utilisera, se présente ainsi :

```
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 begin_pan(Point2D start) { ... }
      public void pan(double dx, double dy) { ... }
      public void end_pan() { ... }

      public void begin_zoom(Point2D start) { ... }
      public void zoom(double dx, double dy) { ... }
      public void end_zoom() { ... }
}
```

Les `MouseEvent e` ont 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 ;
- `String e.getEventType().getName()` qui retourne le type d'évènement dans une chaîne de caractères : `"MOUSE_PRESSED"` pour ↓, `"MOUSE_RELEASED"` pour un ↑ et `"MOUSE_DRAGGED"` 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 7

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.