Interaction Homme-Machine

durée indicative : **2 heures**
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 (6 points)

Question 1

Bob utilise un éditeur de texte. Il vient de sélectionner un passage avec la souris et souhaite le passer en gras. Deux possibilités s'offrent à lui : 1) aller cliquer dans la barre de menu pour ouvrir le menu «Format», puis sur l'item «Police» pour ouvrir un sous-menu, et enfin sur la commande «Gras» de ce sous-menu ; ou bien 2) invoquer cette commande grâce au raccourcit «cmd-B» après avoir ramené sa main sur le clavier.

a) En utilisant le modèle KLM (cf. valeurs en annexe), comparez ces deux suites d'actions.
b) Quelle distance de la théorie de l'action de Norman est mesurée par KLM ?

Question 2

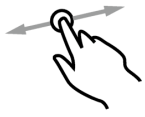
a) Comment distinguer ce qui fait partie du noyau fonctionnel de ce qui fait partie de l'interface d'une application ?
b) Selon Fekete [1996], le noyau fonctionnel doit fournir les trois services suivants : notification, prévention des erreurs, et annulation. Expliquez pourquoi en donnant un exemple d'interaction impossible ou difficile à réaliser pour chacun de ces services lorsqu'il est absent.

Programmation par événements (8 points)

Question 3

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

- avec 1 doigt, la photo est déplacée (déplacement géré par les fonctions données suivantes : `start_drag(Point start), drag(int dx, int dy), end_drag()`) ;



- avec 2 doigts, la photo est déplacée/tournée/zoomée (redimensionnement géré par la fonction donnée suivante : `pinch(Point old_p1, Point p1, Point old_p2, Point 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).

Dessinez la machine à état qui réalise ces interactions (vous pouvez faire l'hypothèse qu'au plus 2 doigts seront utilisés en même temps).

Question 4
Réalisez en java la machine à état de la question précédente.
Pour cela, vous est donnée la classe `Surface` qui utilisera votre classe `Listener` de la manière suivante :

```
package exam.ui;

import javax.swing.JComponent;
import java.awt.Point;

public class Surface extends JComponent {
   Listener listener;
   public Surface() {
      listener = new Listener(this);
      addMouseListener(listener);
      addMouseMotionListener(listener);
      ...
   }

   public void start_drag(Point start) { ... }
   public void drag(int dx, int dy) { ... }
   public void end_drag() { ... }

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

Votre classe `Listener` pourra commencer ainsi :

```
package exam.ui;

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

Les doigts génèrent en effet des événements souris (`MouseEvent`) qui ont simplement une méthode supplémentaire pour connaître l'identifiant du doigt correspondant.
Les `MouseEvent` ont ainsi les méthodes utiles suivantes :
- `Point getPoint()` qui retourne le lieu de l'événement ;

- `int getX()` et `int getY()` qui retournent la même information ;
- `int 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.

Les interfaces fournies par `java.awt.event` permettant de recevoir les événements sont données ci-dessous, elles disposent toutes d'un adaptateur associé (`XxxxAdapter` pour `XxxxListener`).

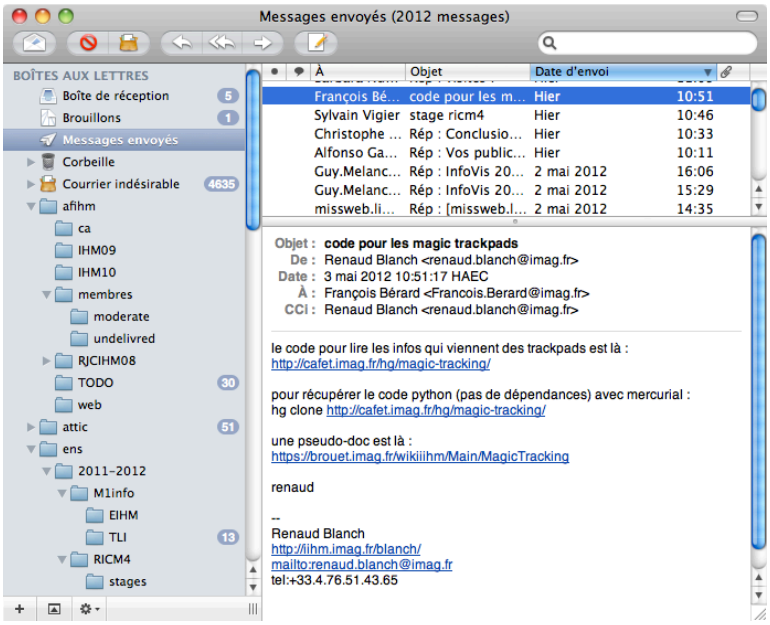
```
public interface MouseListener {
      public void mousePressed(MouseEvent e);
      public void mouseReleased(MouseEvent e);
      ...
}

public interface MouseMotionListener {
      public void mouseDragged(MouseEvent e);
      ...
}
```

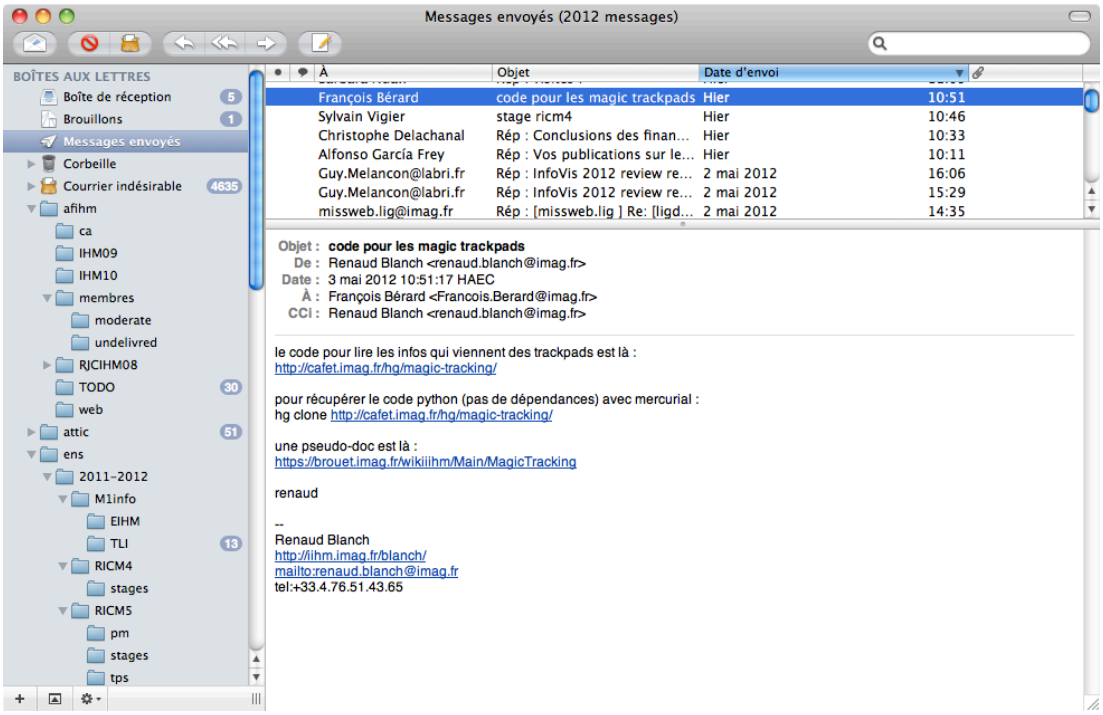
L'interface `MouseInputListener` qui réunit ces deux interfaces est également disponible.

Composants interactifs (4 points)

Question 5

Les figures ci-dessous donnent la fenêtre principale de l'application Mail de Mac OS X avant et après un redimensionnement. Sachant que la largeur de la colonne de gauche qui contient les boîtes aux lettres et 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 Java/SWING en spécifiant si nécessaire pour les conteneurs les gestionnaires de géométrie utilisés et la localisation de leurs fils.
Si certains aspects ne peuvent être rendu avec SWING (e.g., les boutons collés dans la barre de boutons), simplifiez.





Annexes

Théorie de l'action de Norman

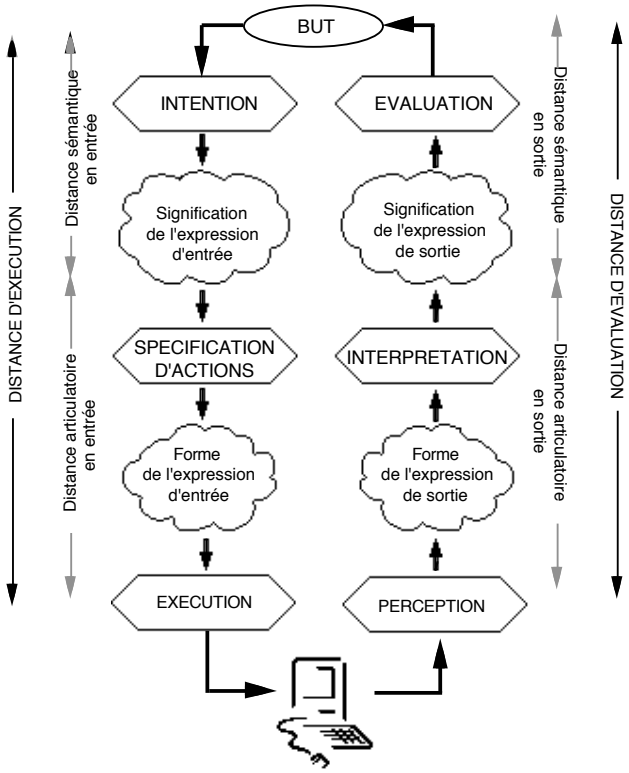


Fig. 3.3 : *Distances sémantiques et distances articulatoires.*

Keystroke Level Model

Les opérateurs de KLM sont :

K	frappe d'un caractère	.20 s
P	pointage à la souris	1.10 s
H	déplacement de la main de la souris au clavier	.40 s
M	réflexion	1.35 s