Interaction Homme-Machine

durée indicative : **1h30 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 (4 points)

Question 1

Pour supprimer un fichier l'utilisateur Bob utilise un terminal. Son interaction se passe ainsi (en gras les commandes entrées par Bob) :

```
% c temp/exam/
c: Command not found.
% cd temp/exam/
% ls
titi.txt    toto.txt
% rm titi.txt
% ls
toto.txt
%
```

a) Expliquez en quoi il ne s'agit pas de manipulation directe en détaillant les principes de celle-ci qui ne sont pas vérifiés dans cet exemple.
b) Du point de vue de la théorie de l'action de Norman (cf. schéma en annexe), comparez les différentes distances en jeu pour deux utilisateurs familiers avec les ordinateurs, l'un étant habitué à utiliser la ligne de commande, l'autre non.

Programmation par événements (8 points)

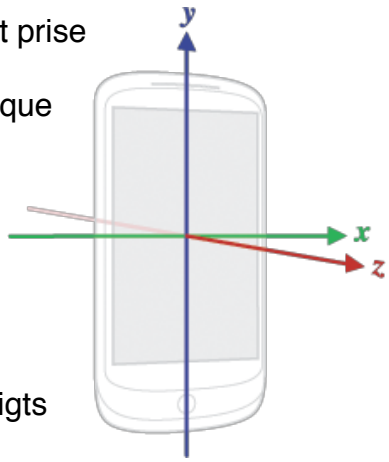
Question 2

a) Expliquez pourquoi il faut séparer le code du noyau fonctionnel de celui de l'interface.
b) Expliquez pourquoi le mécanisme des *callbacks* (fonctions de rappel) est important dans cette séparation.

Question 3

Un des problèmes des dispositifs mobiles à écran tactile (tablettes, téléphones, etc.) est l'absence du mode «survol» du pointeur : alors qu'avec la souris on peut positionner le curseur sur un objet sans forcement enfoncer de bouton, le tactile ne permet pas de survoler un objet lorsqu'il n'y a pas contact avec l'écran. Cette limitation fait qu'il n'existe pas pour les mouvements avec le tactile de distinction équivalente à celle qui existe entre les mouvements avec bouton enfoncé *(drag)* et sans bouton enfoncé *(motion)* pour la souris.

Pour pallier ce manque Scoditti a proposé la technique TouchOver qui utilise l'inclinaison du dispositif (angle de rotation autour de l'axe des X lorsqu'il est en orientation portrait ; cf. schéma) pour simuler un bouton :



- quand le doigt est posé sur l'écran, l'inclinaison du mobile est prise comme référence ;
- lorsque le doigt bouge, l'application se comporte comme lorsque la souris bouge sans bouton enfoncé ;
- lorsque l'inclinaison dépasse de plus de 10 degrés l'inclinaison initiale, l'application se comporte comme si le bouton gauche venait d'être enfoncé ;
- lorsque l'inclinaison revient à une valeur inférieure à celle mesurée initialement, l'application se comporte comme si le bouton venait d'être relâché ; et
- entre ces deux derniers évènements, les mouvements de doigts sont interprétés comme un déplacement de souris, bouton gauche enfoncé (e.g. un *drag*).

Pour les questions suivantes, on considérera que l'on est toujours en mode portrait et que l'on utilise qu'un doigt à la fois.

a) En considérant les évènements : ↓ (doigt qui arrive en contact de l'écran) ; ↑ (doigt qui repart) ; M (doigt qui bouge sur l'écran) ; et RX (rotation du mobile autour de son axe des X), dessinez la machine à états qui déclenche les actions suivantes selon la spécification informelle donnée ci-dessus : `move(Point p)` ; `drag(Point p)` ; `press(Point p)` ; et `release(Point p)` (resp. curseur qui bouge sans/avec bouton enfoncé ; bouton gauche enfoncé/relâché).

Les évènements ↓, ↑ et M ont une méthode `Point getPoint()` qui donne la position du doigt sur l'écran et l'évènement RX une méthode `int getPitch()` qui donne l'angle du mobile autour de l'axe des X en degrés.

b) Réalisez en java cette machine à états, en considérant que les évènements des doigts et de l'inclinaison sont gérés grâce aux interfaces `FingerListener` et `OrientationListener` décrites ci-dessous (elles disposent chacune d'un adaptateur associé `XxxxAdapter` pour `XxxxListener`) :

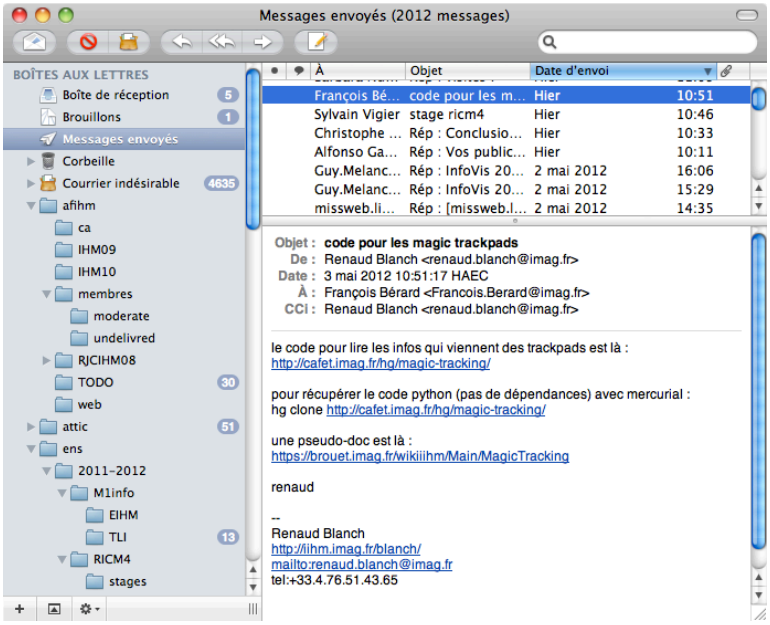
```
public interface FingerListener {
      public void fingerPressed(FingerEvent e);
      public void fingerReleased(FingerEvent e);
      public void fingerMoved(FingerEvent e);
}

public interface OrientationListener {
      public void orientationChanged(OrientationEvent e);
}
```

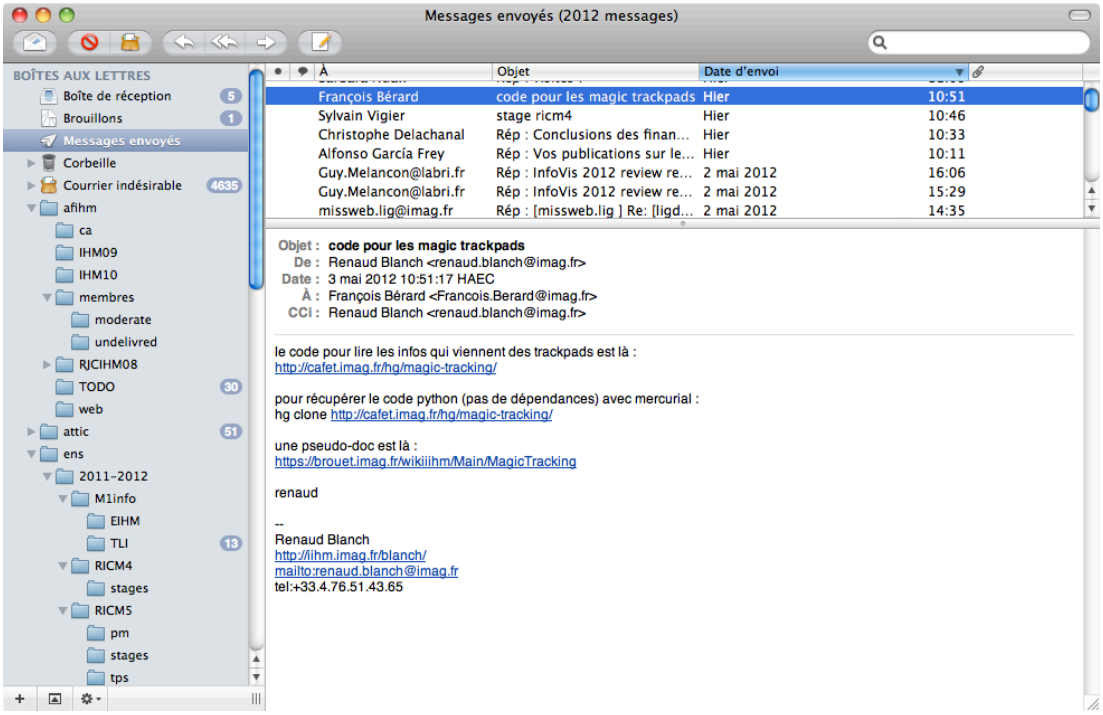
Les `FingerEvent` ont une méthode `Point getPoint()` et les `OrientationEvent` ont une méthode `int getPitch()`.

Composants interactifs (6 points)

Question 4

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. La liste des messages est simplement une table, et le message lui-même est un JTextPane.





Annexes

Théorie de l'action de Norman

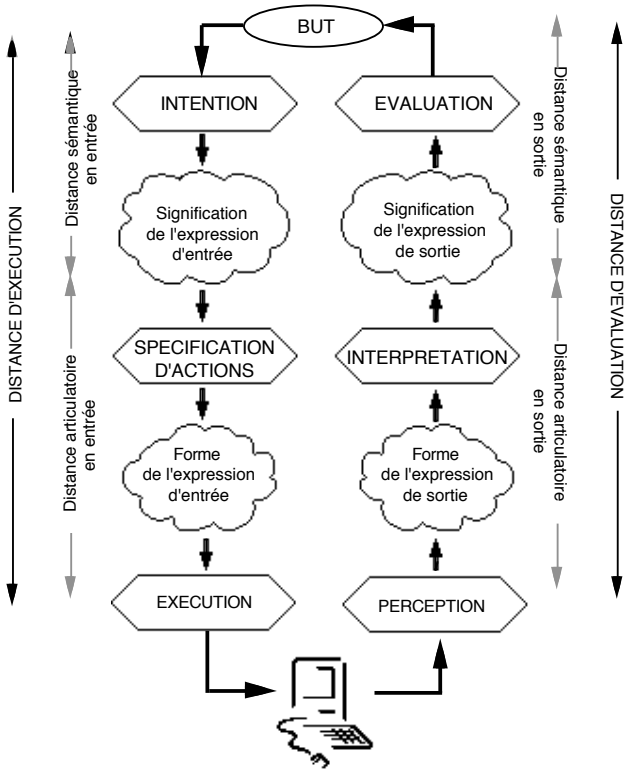


Fig. 3.3 : *Distances sémantiques et distances articulatoires.*