Techniques des Logiciels Interactifs

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 (7 points)

Question 1
Il existe deux sortes de boîtes à outils de construction d'interface : celles qui ont une approche minimaliste et celles qui suivent l'approche maximaliste.
a) Expliquez en une phrase ce qui les distingue.
b) Donnez les avantages et les inconvénients de ces deux approches.
c) Donnez le nom d'une boîte à outils de chaque sorte.

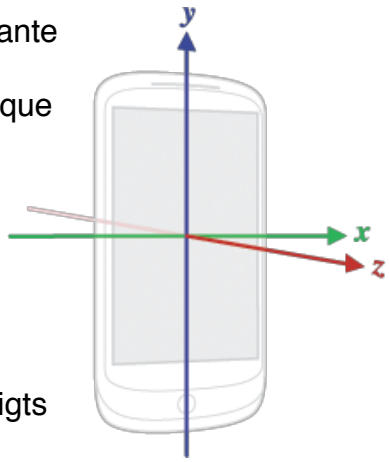
Question 2
Selon le modèle de Seeheim, le code de l'interface utilisateur peut se décomposer en trois parties : la présentation, le contrôleur de dialogue et l'interface du noyau fonctionnel, correspondant respectivement aux niveaux lexical, syntaxique et sémantique de l'interaction.
a) Dans un explorateur de fichiers, on utilise la souris pour glisser l'icone d'un fichier sur celle d'un répertoire pour y déplacer le fichier. Expliquez à quoi correspondent les trois niveaux ci-dessus pour l'interaction en entrée à la souris sur cet exemple.
b) Le(s)quel(s) de ce(s) niveau(x) est/sont fourni(s) par les boîtes à outils de construction d'interface ?

Question 3
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.

Programmation par événements (8 points)

Question 4
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 forcément enfoncer de bouton, le tactile ne permet pas de survoler un objet lorsqu'il n'y a pas contact avec l'écran. Pour les mouvements avec le tactile il n'existe donc pas de distinction équivalente à celle qui existe pour la souris entre les mouvements avec bouton enfoncé *(drag)* et sans bouton enfoncé *(motion)*.

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 :



- au moment où le doigt est posé sur l'écran, l'inclinaison courante 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 n'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 :

```
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 (5 points)

Question 5

Les figures ci-dessous donnent la fenêtre principale de l'application Mail de Mac OS X avant et après un redimensionnement. 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'arbre des widgets Java/SWING pour cette interface. Spécifiez si nécessaire les gestionnaires de géométrie utilisés et la localisation des fils. Certains aspects ne peuvent être rendus avec SWING (e.g., les boutons collés dans la barre de boutons), simplifiez.
Ne détaillez pas la liste des boîtes aux lettres, la liste des messages et l'affichage du message courant.