Introduction à l'Interaction Homme-Machine

durée de l'épreuve : 3 heures

durée indicative de la partie : 1 heure 30 min

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.

Partie I : Principes et outils pour les logiciels interactifs (10 points)

Question 1 (3 points)

- 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 2 (3 points)

La figure de droite présente les différentes étapes de la réalisation d'une tâche d'après la théorie de l'action [Norman, 1986].

Soit le scénario suivant :

j'édite un document à l'aide d'un logiciel de traitement de texte, je souhaite mettre un passage sélectionné en gras.

- a) Explicitez les différentes étapes qui permettront d'effectuer cette tâche d'après Norman pour deux réalisations possibles : utiliser un raccourcit clavier et utiliser un menu de l'application.
- b) Donnez dans chaque cas les différentes distances mises en jeu.

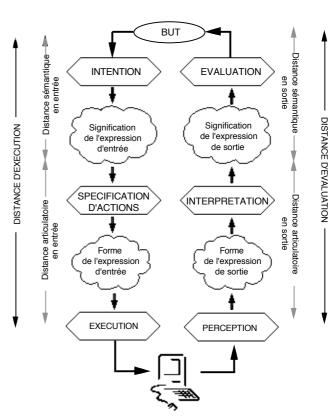


Fig. 3.3 : Distances sémantiques et distances articulatoires.

Question 3 (4 points)

Soit le code suivant qui réalise l'interaction de translation lors du déplacement de la souris, bouton gauche enfoncé, et celle de zoom sur le rectangle défini par les points où le bouton droit a été enfoncé puis relâché après déplacement de la souris.

```
package grapher.ui;
. . .
import java.awt.Point;
import java.awt.Cursor;
import java.awt.event.*;
import javax.swing.event.*;
import javax.swing.JPanel;
public class GraphPresentation extends JPanel {
      protected Listener listener;
      public GraphPresentation() {
            listener = new Listener();
            addMouseListener(listener);
            addMouseMotionListener(listener);
      }
      . . .
      private class Listener
            extends MouseInputAdapter {
            final Cursor hand cursor = new Cursor(Cursor.HAND CURSOR);
            final Cursor default_cursor = Cursor.getDefaultCursor();
            Point mouse, left, right;
            public void mousePressed(MouseEvent e) {
                  mouse = e.getPoint();
                  switch(e.getButton()) {
                  case MouseEvent.BUTTON1:
                        left = mouse;
                        setCursor(hand_cursor);
                        break;
                  case MouseEvent.BUTTON3:
                        right = mouse;
                        break;
                  }
            }
```

```
public void mouseReleased(MouseEvent e) {
                  switch(e.getButton()) {
                  case MouseEvent.BUTTON1:
                        left = null;
                        setCursor(default_cursor);
                        break;
                  case MouseEvent.BUTTON3:
                        zoom(right, mouse);
                        right = null;
                        repaint();
                        break;
                  }
            }
            public void mouseDragged(MouseEvent e) {
                  mouse = e.getPoint();
                  if(left != null) {
                        translate(mouse.x - left.x, mouse.y - left.y);
                        left = mouse;
                  if(right != null) {
                        repaint();
                  }
            }
      }
      protected void translate(int dX, int dY) { ... }
      protected void zoom(Point p0, Point p1) { ... }
}
```

- a) Dessinez la machine à états de l'interaction réalisée.
- b) Modifiez la réalisation de la classe Listener pour qu'elle utilise une énumération pour stocker son état courant.