Projet de S.D. :
Librairie de contrôle à Distance d'une application

1 Travail demandé

Le travail demandé comporte une modélisation et une implantation en Java, C, C++ ou Lisaac (au choix). Un rapport court (2 pages maximum) devra accompagner ce projet pour décrire les choix de votre implantation en format papier.

Implantation

- Votre librairie devra utiliser une technologie de programmation distribuée vue en cours, à savoir : RPC, RMI, Corba (ou Soap).
- Un petit programme d'exemple mettra en lumière l'utilisation de votre librairie.
- L'ensemble devra fonctionner sur la machine *turing* de l'UDS.

2 Remise du projet

La remise et l'évaluation du projet se fera la semaine du **09/05**. Les sources seront envoyées, la même semaine.

Une soutenance sera organisée.

3 Réalisation du projet

- La réalisation de ce projet devra se faire impérativement à deux ou seul.
- Comme tout cahier des charges, celui-ci ne peut être exhaustif. En cas d'ambiguïté, précisez votre interprétation personnelle. Toute solution cohérente, justifiée et non contradictoire avec ce cahier des charges sera acceptée.

4 Sujet

De nombreux outils sont disponibles sur internet pour réaliser le contrôle d'un ordinateur à distance. Ces outils se situent au niveau du système d'exploitation et nécessitent que les deux ordinateurs possèdent le même système d'exploitation et le même logiciel de contrôle. Ainsi, le problème actuel est qu'un ordinateur sous un certain système d'exploitation ne peut prendre le contrôle d'un ordinateur ayant un autre système d'exploitation. Il n'existe pas, à ma connaîssance, une librairie permettant de réaliser ce contrôle simplement au sein d'une application donnée. Cette approche permettrait de se libérer de la contrainte des systèmes d'exploitation et d'apporter le contrôle d'une application multi-plateforme (écrit en Java par exemple) sur des systèmes d'exploitation diffèrents. Dans la pratique, ce contrôle peut être utile pour montrer à un client les fonctionnalités d'une application ou de le dépanner à distance sans l'intégration de logiciel extérieur et sans la contrainte liée au système d'exploitation.

Simplification du problème

Nous considérons la représentation graphique d'une application identique sous n'importe quel système d'exploitation. Nous considérons aussi la taille de la fenêtre identique entre les deux machines. Nous nous intéressons simplement au contrôle de la souris.

Avec ces considérations, une action souris à une coordonnée précise doit réaliser la même tâche entre les deux machines. Donc, l'objectif est de faire transiter des actions souris entre deux machines de manière transparente.

Le système doit être *full duplex*, c'est à dire qu'une action physique d'une machine doit être répercutée sur l'autre ordinateur.

Intégration dans l'application

L'intégration de notre librairie lors de la conception d'une application doit être la plus transparente possible. Deux méthodes doivent suffire :

- `connect_gui()` : Permettant d'établir la connexion entre deux ordinateurs. Cette fonction doit prendre en paramètre tout ce qui est nécessaire pour cette connexion.
- `deconnect_gui()` : Permet de rompre la connexion. Un message sera envoyé à l'autre ordinateur pour signaler cette déconnexion.

Evolution possible

Voici quelques idées d'amélioration de votre application :

- Prise en compte d'autres événements comme le clavier.
- Adaptation dynamique de la taille de la fenêtre entre les deux machines.

5 Exemples

Les exemples ci-dessous permettent de récupérer les événements souris et clavier de manière transparente. Aussi, nous simulons le click de la souris lors de la saisie d'une touche. Ces exemples sont simplement là pour vous donner une piste, tout autre manière de faire reste possible.

Java : `IHMSimpliste.java`

```
import java.awt.*;
import java.awt.event.*;
// Application normale.
public class IHMSimpliste extends Frame {
 public IHMSimpliste() {
  Label  l = new Label(
     "Press Key for Click simulation");
  Button b1 = new Button("Exit");
  Button b2 = new Button("Other");
  b1.addActionListener(
     new ActionListener() {
 public void actionPerformed(ActionEvent e)
 { System.exit(0); } });
  setLayout(new GridLayout(0, 1));
  add(l);
  add(b1);
  add(b2);
 }
 public static void main(String args[]) {
```

```
  Frame f;
  Espion e;
  f=new IHMSimpliste();
  // Utilisation de notre librairie.
  e = new Espion(f);

  f.pack();
  f.setSize(220+10,80+29);
  f.setVisible(true);
 }
}
// Notre librairie.
class Espion implements
   MouseListener,KeyListener {
 private Frame frame;
 private Robot robot;
 public Espion(Frame f) {
  super();
  frame = f;
  // Robot pour simuler le click.
  try {
    robot = new Robot();
  } catch (java.awt.AWTException e) { };
  int j;
  for (j=0;j<f.getComponentCount();j++) {
    f.getComponent(j).addMouseListener(this);
    f.getComponent(j).addKeyListener(this);
  };
 };
 public void mouseClicked(MouseEvent e) { }
 public void mouseEntered(MouseEvent e) { }
 public void mouseExited(MouseEvent e)  { }
 public void mousePressed(MouseEvent e) {
  System.out.println("Press!");
 }
 public void mouseReleased(MouseEvent e) {
  System.out.println("Release!");
 }
 public void keyPressed(KeyEvent e) { }
 public void keyReleased(KeyEvent e) {
  // Simulation d'un click de souris.
  robot.mousePress(InputEvent.BUTTON1_MASK);
  robot.mouseRelease(InputEvent.BUTTON1_MASK);
 }
 public void keyTyped(KeyEvent e)  { }
};
```

Lisaac : `example.li`

```
Section Header
  + name       := EXAMPLE;
Section Inherit
  - parent_inbox:INBOX := INBOX;
Section Public
  // Notre Librairie
  - init_espion <-
  ( MOUSE.add_client Self;
    KEYBOARD.add_client Self;
  );
  - receive msg:EVENT <-
  ( + mouse:EVENT_MOUSE;
    + keyb:EVENT_KEYBOARD;
    mouse ?= msg;
    (mouse != NULL).if {
      ((mouse.left_up) || {mouse.right_up}).if {
        "Press!".println;
      }.elseif {(mouse.left_down) ||
            {mouse.right_down}} then {
        "Release!".println;
      };
    };
    keyb ?= msg;
    (keyb != NULL).if {
      // Simulation d'un click de souris.
      MOUSE.set
      (MOUSE.x_current,MOUSE.y_current)
      with (TRUE,FALSE);
      MOUSE.acknowledge;
      MOUSE.set
      (MOUSE.x_current,MOUSE.y_current)
      with (FALSE,FALSE);
    };
  );
  // Application Normale.
  - main <-
  ( VIDEO.make (220,80);
    INTERFACE.make VIDEO size
      (VIDEO.width,VIDEO.height) with (
      G_RAW.create
        (G_OUT.create
        "Press Key for Click simulation"
        ).fix_height /
      G_BUTTON.create (G_OUT.create "Exit")
        action { b:G_BUTTON;
          die_with_code 0; } /
      G_BUTTON.create (G_OUT.create "Other")
    );
    // Utilisation de notre librairie.
    init_espion;

    INTERFACE.run;
  );
```

Good luck!