Diseño de Interfaces de Usuario

Nombre: Zabai Armas Herrera

Curso: 4º

Índice:

Práctica 6. Visor de imágenes avanzado

Uso del Viewport, Dragging, Listeners y Menus

Base:

1. Creación de elementos de la interfaz necesarios para ver la imagen y los datos de la misma.

Ilustración 1 Elementos de información para la imagen

2. Configuración para asignar los campos de texto.

```
public MainFrame() {
    initComponents();
    setLocationRelativeTo(null);

    Dimension iDimension = Image.getSize();
    iWidth = iDimension.width;
    iHeight = iDimension.height;
    IWidth.setText(String.valueOf(iWidth));
    IHeight.setText(String.valueOf(iHeight));
```

Ilustración 2 Asignar la información a los campos de texto

3. Creación de los componentes para la disposición de información del cursor.

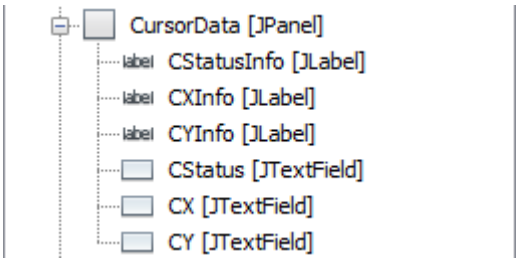


Ilustración 3 Elementos de información para el cursor

4. Actualización de los estados del cursor.

```
private void ImageMousePressed(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    Image.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    CStatus.setText("Presionado");
```

Ilustración 4 Evento al presionar el ratón

```
private void ImageMouseReleased(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    Image.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
    CStatus.setText("Liberado");
}
```

Ilustración 5 Evento al soltar el ratón

5. Actualización de la posición del cursor.

```
private void ImageMouseDragged(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    CStatus.setText("Arrastrando");

    Point position = Image.getMousePosition();
    cX = position.x;
    cY = position.y;
    CX.setText(String.valueOf(cX));
    CY.setText(String.valueOf(cY));
```

Ilustración 6 Actualizar cuando se hace "drag"

```
private void ImageMouseMoved(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    Point position = Image.getMousePosition();
    cX = position.x;
    cY = position.y;
    CX.setText(String.valueOf(cX));
    CY.setText(String.valueOf(cY));
}
```

Ilustración 7 Actualizar al mover el ratón

6. Creación de elementos para ver la información del Viewport.

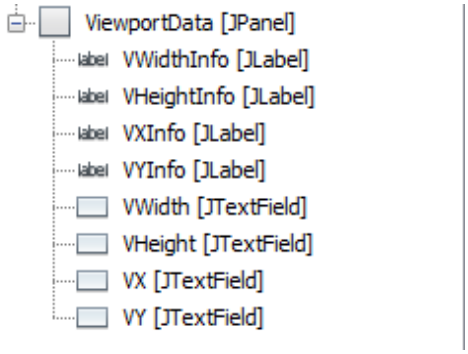


Ilustración 8 Elementos de información para el Viewport

7. Con todo esto preparado, sólo falta la programación para que muestre la información y cumpla la funcionalidad de mover la imagen.

```
view = Scroll.getViewport();
Dimension vDimension = view.getExtentSize();
vWidth = vDimension.width;
vHeight = vDimension.height;
VWidth.setText(String.valueOf(vWidth));
VHeight.setText(String.valueOf(vHeight));
Point p = view.getViewPosition();
vX = p.x;
vY = p.y;
VX.setText(String.valueOf(vX));
VY.setText(String.valueOf(vY));

// Listener asociados a las barras de Scroll de forma manual
hBar = Scroll.getHorizontalScrollBar();
vBar = Scroll.getVerticalScrollBar();
hBar.addAdjustmentListener(new java.awt.event.AdjustmentListener() {
    @Override
    public void adjustmentValueChanged(AdjustmentEvent e) {
        hScrollAdjust(e);
    }
});
vBar.addAdjustmentListener(new java.awt.event.AdjustmentListener() {
    @Override
    public void adjustmentValueChanged(AdjustmentEvent e) {
        vScrollAdjust(e);
    }
});
```

Ilustración 9 Asigna información a los campos de texto y crea Listeners manuales

```
private void hScrollAdjust(java.awt.event.AdjustmentEvent e){
    Point p = view.getViewPosition();
    vX = p.x;
    VX.setText(String.valueOf(vX));
}

private void vScrollAdjust(java.awt.event.AdjustmentEvent e){
    Point p = view.getViewPosition();
    vY = p.y;
    VY.setText(String.valueOf(vY));
}
```

Ilustración 10 Listeners

```java
private void ImageMousePressed(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    Image.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
    CStatus.setText("Presionado");

    Point position = Image.getMousePosition();
    cX0 = position.x;
    cY0 = position.y;
    p0 = view.getViewPosition();
}
```

Ilustración 11 Cuando pulsamos guardamos la coordenadas del ratón

```java
private void ImageMouseDragged(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    CStatus.setText("Arrastrando");

    Point position = Image.getMousePosition();
    cX = position.x;
    cY = position.y;
    CX.setText(String.valueOf(cX));
    CY.setText(String.valueOf(cY));

    Point p = view.getViewPosition();
    p.x = p0.x - (int) ((cX-cX0)/1.5);
    p.y = p0.y - (int) ((cY-cY0)/1.5);
    if(p.x < 0) p.x = 0;
    if(p.x > iWidth-vWidth && iWidth > vWidth) p.x = iWidth - vWidth;
    if(p.y < 0) p.y = 0;
    if(p.y > iHeight-vHeight && iHeight > vHeight) p.y = iHeight - vHeight;
    view.setViewPosition(p);
}
```

Ilustración 12 Actualizamos el Viewport cuando arrastramos

8. Entrando en la recta final, vamos con los elementos del menú.

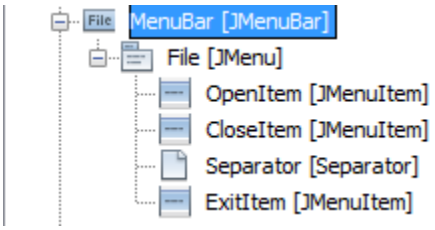


Ilustración 13 Elementos del menú

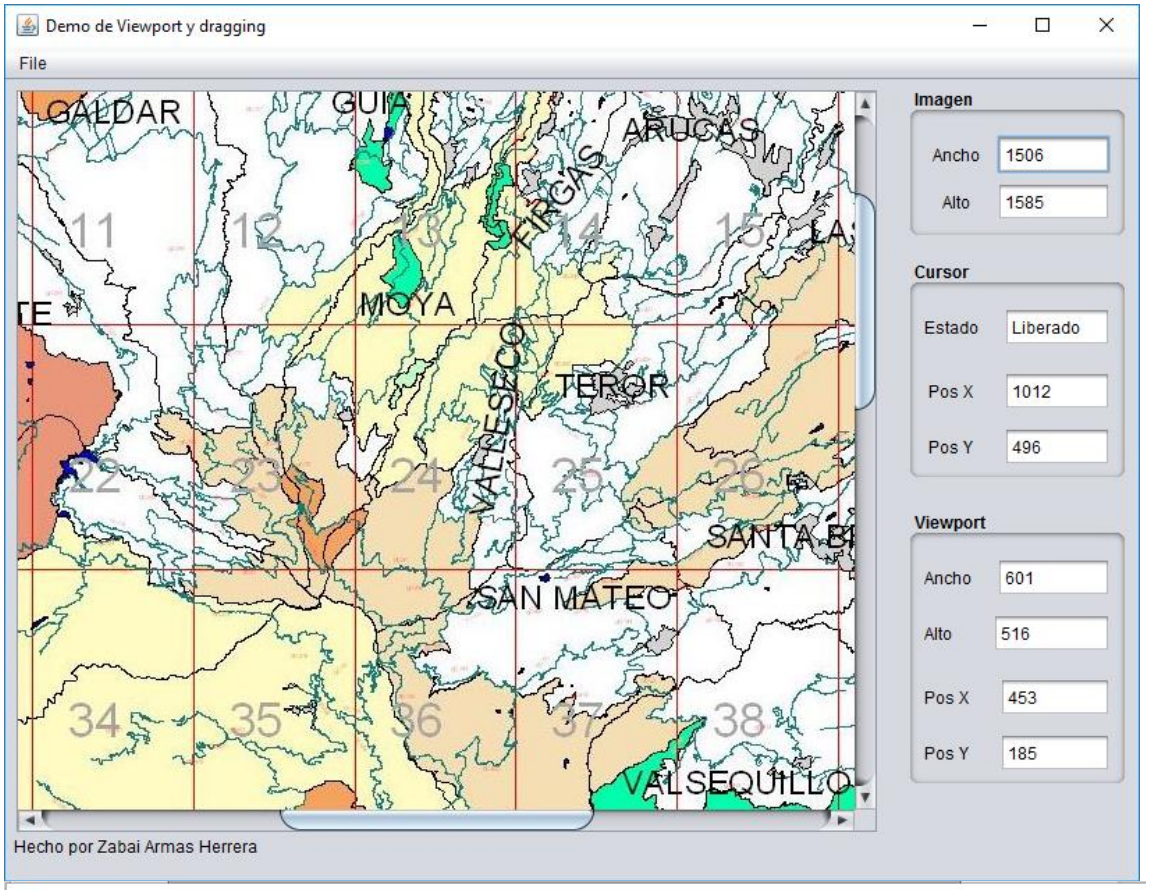
9. Añadiendo las funcionalidades requeridas.

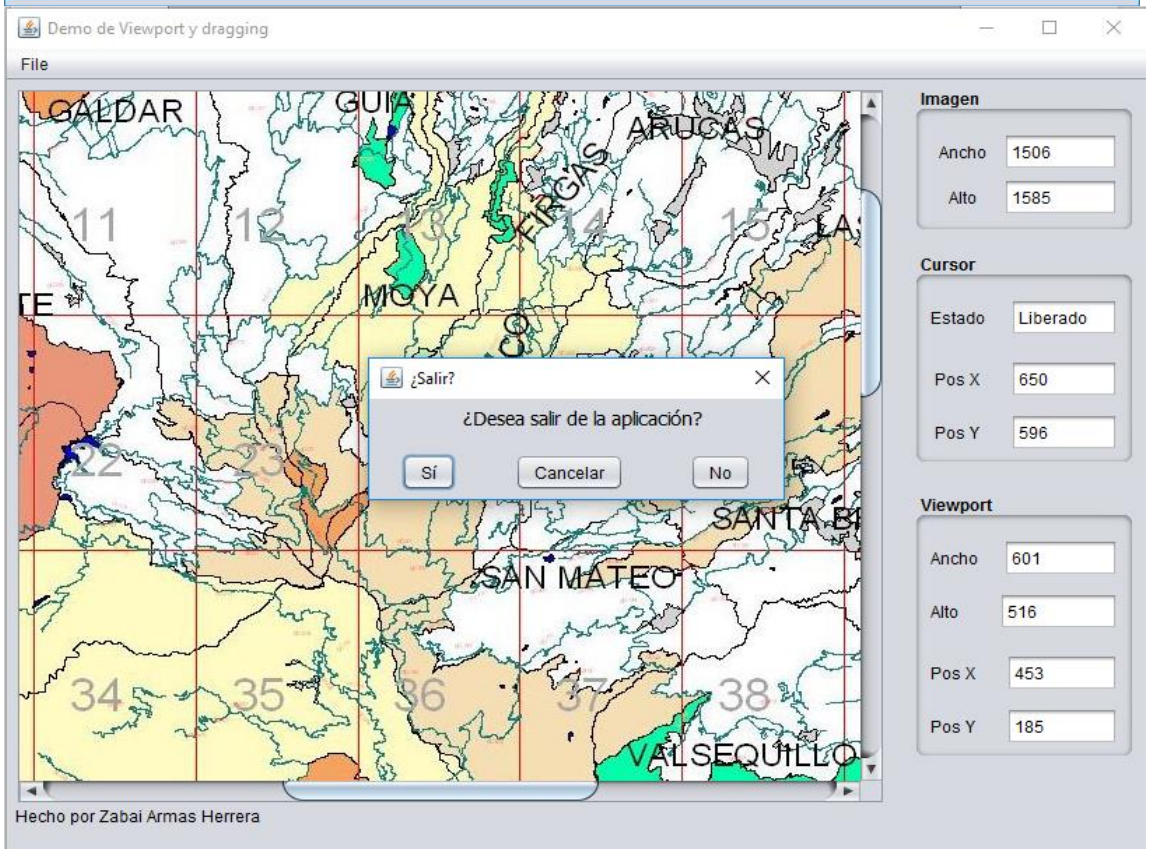
```
private void ExitItemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    ExitDialog exitDialog = new ExitDialog(this, rootPaneCheckingEnabled);
}

private void OpenItemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setCurrentDirectory(null);
    fileChooser.addChoosableFileFilter(new ImageFilter());
    int returned = fileChooser.showOpenDialog(OpenItem);

    if(returned == JFileChooser.APPROVE_OPTION){
        File file = fileChooser.getSelectedFile();
        Image.setIcon(new javax.swing.ImageIcon(file.getAbsolutePath()));
        SwingUtilities.updateComponentTreeUI(this);
    }
}

private void CloseItemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Image.setIcon(null);
}
```

Ilustración 14 Eventos para el menú

Si es necesario la comprobación de código o ejecución del mismo, este proyecto y todas las demás prácticas pueden encontrarse en el repositorio GitHub del autor.

Capturas:


Demo de Viewport y dragging
File
GALDAR
GUIA
ARUCAS
FIRGAS
MOYA
TEROR
SANTA BÁ
SAN MATEO
VALSEQUILLO
Imagen
Ancho 1506
Alto 1585
Cursor
Estado Liberado
Pos X 1012
Pos Y 496
Viewport
Ancho 601
Alto 516
Pos X 453
Pos Y 185
Hecho por Zabai Armas Herrera


Demo de Viewport y dragging
File
¿Salir?
¿Desea salir de la aplicación?
Sí
Cancelar
No
Imagen
Ancho 1506
Alto 1585
Cursor
Estado Liberado
Pos X 650
Pos Y 596
Viewport
Ancho 601
Alto 516
Pos X 453
Pos Y 185
Hecho por Zabai Armas Herrera

Bibliografía:

- PDF de la práctica
- Documentación de Java
    - Eventos del ratón
- Stack Overflow