Application: gvSIG desktop - gvSIG feature requests #540
Añadir nueva funcionalidad a gvSIG: Snapper de seguimiento

04/18/2012 01:28 PM - Leticia Riestra

Status:	Closed	**% Done:**	0%
Priority:	High	**Spent time:**	0.00 hour
Assignee:			
Category:			
Target version:			
gvSIG version:	2.0.0	**Add-on resolve version:**	
Keywords:		**Add-on resolve build:**	
Has patch:	No	**Proyecto:**	
Add-on name:	Unknown	**Hito:**	
Add-on version:			

Description

Hola
Desde el laboratorio de Bases de Datos de A Coruña nos gustaría proponeros una nueva funcionalidad que no tenéis hecha y que consideramos que es bastante útil: hacer snapper de seguimiento.
El snapper de seguimiento lo que permite es hacer un snapper que va siguiendo la geometría. De esa forma, cuando el usuario está digitalizando una geometría y desea que vaya justo pegada a otra geometría adyacente, no tiene que indicar tantos puntos y estará seguro al 100% de que la geometría limita perfectamente con la otra que está digitalizando puesto que el snapper se encarga de detectar todos los puntos intermedios entre los que el usuario haya indicado (espero haber dejado claro el funcionamiento porque no estoy muy segura de haberme explicado muy bien).

Es decir, que funciona tal y como ya funcionaban los snapper "FinalPointSnapper" y "NearestPointSnapper" pero a mayores le añade un mejor funcionamiento al permitir seguir la geometría de aquello que esté más cerca.
Además, damos la posibilidad de desactivar o activar los snappers. Cuando la capa está en edición, se mostrará un botón que permitirá al usuario desactivar/activar los snappers

Para ello, habría que realizar las siguientes acciones:

- Generar dos clases dentro del proyecto *libFMap_controls* en el paquete *org.gvsig.fmap.mapcontrol.tools.snapping.snappers*

Interfaz ISnapperSeguimiento

```
package org.gvsig.fmap.mapcontrol.tools.snapping.snappers;

import java.awt.geom.Point2D;
import java.util.ArrayList;

/**
 * Interfaz con la declaracion del método necesario para poder hacer el snapping de seguimiento
 *
 * @author Leticia Riestra Ainsua
 *
 */
public interface ISnapperSeguimiento extends ISnapper{

    //Devolverá un ArrayList con los puntos necesarios para el snapping de seguimiento
    ArrayList<Point2D> getSnappedPoints();

}
```

Interfaz ISnapperVectorialSeguimiento

```
package org.gvsig.fmap.mapcontrol.tools.snapping.snappers;

import java.awt.geom.Point2D;
import org.gvsig.fmap.geom.Geometry;

/**
 *
 * @author Leticia Riestra
 *
 */
public interface ISnapperVectorialSeguimiento extends ISnapperSeguimiento {

    Point2D getSnapPoint(Point2D queryPoint, Geometry geomToSnap, double tolerance, Point2D lastPointEntered);

}
```

- Añadir a la clase *MapControl* (*org.gvsig.fmap.mapcontrol*) los siguientes métodos

```
/**
     * Método creado debido al snapping. Usado para recuperar el usedSnap que se utilizará
     * en CADToolAdapter
     * @return ISnapperSeguimiento usedSnap
     */
    public ISnapper getUsedSnap(){
        return usedSnap;
    }

    /**
     * Método para establecer el valor de la variable previousPoint. Se usará por CADToolAdapter
     * para poder establecer el valor de previousPoint que se usará en este clase (de la otra
     * forma dicho valor nunca se actualizaba y siempre valía null)
     * @param previous
     */
    public void setPreviousPoint(double [] previous){
        previousPoint = previous;
    }
```

- Modificar el método adjustToHandler de MapControl (línea 2538 aprox.) añadiendo el siguiente código (entre asteriscos se indica el código a añadir)

```
if (theSnapper instanceof ISnapperRaster)
{
    ISnapperRaster snapRaster = (ISnapperRaster) theSnapper;
    theSnappedPoint = snapRaster.getSnapPoint(this, point, mapTolerance, lastPoint);
}
```

```
*if (theSnapper instanceof ISnapperVectorialSeguimiento)
{
    theSnappedPoint = ((ISnapperVectorialSeguimiento) theSnapper).getSnapPoint(point, geom, mapTolerance, lastPoint);
}*

if (theSnappedPoint != null) {
...
```

- Crear dentro del proyecto *extEditing* un paquete llamado *org.gvsig.editing.gui.cad.snapping* con dentro dos clases:
  - SeguimientoFinalPointSnapper: ver fichero adjunto SeguimientoFinalPointSnapper
  - SeguimientoNearestPointSnapper: ver fichero adjunto SeguimientoNearestPointSnapper

Nota: El valor 0.000001 indicado dentro de estas dos clases es el umbral de cercanía al que el snapper de seguimiento funciona. Una mejor implementación es con una constante global que substituya dicho valor y así solo hay que cambiar dicha variable global en lugar de estos valores (en nuestro código lo tenemos con una variable global pero aquí os lo hemos indicado con el valor que nosotros le hemos dado)

- Modificar la clase *SnapConfigPage* dentro del proyecto *extEditing* en el paquete *org.gvsig.editing.gui.preferences*
  Hay que cambiar el método initializeDefaults para indicarle que use los nuevos snappers en lugar de los antiguos

```
public void initializeDefaults() {
        for (int n = 0; n < mapControlManager.getSnapperCount(); n++) {
            ISnapper snp = mapControlManager.getSnapperAt(n);
            String nameClass=snp.getClass().getName();
            nameClass=nameClass.substring(nameClass.lastIndexOf('.'));
            if (nameClass.equals(".SeguimientoFinalPointSnapper")){
                int priority = 1;
                prefs.putInt("snapper_priority" + nameClass, priority);
                snp.setEnabled(true);
                snp.setPriority(priority);
            }else if (nameClass.equals(".SeguimientoNearestPointSnapper")){
                int priority = 2;
                prefs.putInt("snapper_priority" + nameClass, priority);
                snp.setEnabled(true);
                snp.setPriority(priority);
            }else if (nameClass.equals(".FinalPointSnapper")){
                int priority = 3;
                prefs.putInt("snapper_priority" + nameClass, priority);
                snp.setEnabled(false);
                snp.setPriority(priority);
            }else if (nameClass.equals(".NearestPointSnapper")){
                int priority = 4;
                prefs.putInt("snapper_priority" + nameClass, priority);
                snp.setEnabled(false);
                snp.setPriority(priority);
            }else{
                int priority = 5;
                prefs.putInt("snapper_priority" + nameClass, priority);
                snp.setEnabled(false);
                snp.setPriority(priority);
            }
```

```
            }
            applySnappers = true;
            snapConfig.setApplySnappers(applySnappers);
            snapConfig.selectSnappers();
        }
```

- Añadir la extensión *SnappersExtension* al proyecto *extEditing* dentro del paquete *org.gvsig.editing* que permitirá registrar la nueva extension

Ver fichero adjunto SnappersExtension

- Modificar la clase *CADToolAdapter* para que tengo los aspectos relacionados con nuestro snapper (debido a que hay que guardar los puntos próximos)

- Cambio 1: declarar el arrayList que contendrá los puntos

```
    /**
    * ArrayList con los Point2D que devuelve el snapper de seguimiento
    */
    private ArrayList<Point2D> otherMapaAdjustedPoints;
```

- Cambio 2: redefinir el método mouseMoved

```
    public void mouseMoved(MouseEvent e) throws BehaviorException {

            lastX = e.getX();
            lastY = e.getY();
            adjustedPoint = e.getPoint();

            // calculateSnapPoint(e.getPoint());
            mapAdjustedPoint =  getMapControl().getMapAdjustedPoint();

            //Debido al snapping
            if (getMapControl().getUsedSnap() != null){
               if (getMapControl().getUsedSnap() instanceof ISnapperSeguimiento){
                  otherMapaAdjustedPoints = ((ISnapperSeguimiento)getMapControl().getUsedSnap()).getSnappedPoints();
               }else{
                  otherMapaAdjustedPoints = null;
               }
            }else{
               otherMapaAdjustedPoints = null;
            }

            showCoords(e.getPoint());

            getMapControl().repaint();
        }
```

- Cambio 3: redefinir el metodo mouseDragged

```
public void mouseDragged(MouseEvent e) throws BehaviorException {
        lastX = e.getX();
        lastY = e.getY();
        adjustedPoint = e.getPoint();

        // calculateSnapPoint(e.getPoint());
        mapAdjustedPoint =  getMapControl().getMapAdjustedPoint();

        //Debido al snapping
        if (getMapControl().getUsedSnap() != null){
           if (getMapControl().getUsedSnap() instanceof ISnapperSeguimiento){
              otherMapaAdjustedPoints = ((ISnapperSeguimiento)getMapControl().getUsedSnap()).getSnappedPoints();
           }else{
              otherMapaAdjustedPoints = null;
           }
        }else{
           otherMapaAdjustedPoints = null;
        }
     }
```

- Cambio 4: redefinir el método mousePressed

```
public void mousePressed(MouseEvent e) throws BehaviorException {
        if (e.getButton() == MouseEvent.BUTTON1) {

           if (otherMapaAdjustedPoints == null
                 || otherMapaAdjustedPoints.size() == 0
                 || !getCadTool().isMultiTransition()) {

              ViewPort vp = getMapControl().getMapContext().getViewPort();
              Point2D p;

              if (mapAdjustedPoint != null) {
                 p = mapAdjustedPoint;
              } else {
                 p = vp.toMapPoint(adjustedPoint);
              }
              transition(new double[] { p.getX(), p.getY() }, e, ABSOLUTE);

           }else{
              //En este caso tenemos que crear una transición por cada uno de los puntos
              //Se pone el cursor en espera porque podría tardar mucho
              MDIManager manager = PluginServices.getMDIManager();
              manager.setWaitCursor();

              for (int i = 0; i < otherMapaAdjustedPoints.size(); i++) {
                 Point2D punto = (Point2D) otherMapaAdjustedPoints.get(i);
                 transition(new double[] { punto.getX(), punto.getY() }, e,
                       ABSOLUTE);
```

```
                }

                manager.restoreCursor();
            }
        }
    }
```

- Cambio 5: redefinir el método paintComponent

```
    public void paintComponent(MapControlDrawer mapControlDrawer) {
        super.paintComponent(mapControlDrawer);
        if (CADExtension.getCADToolAdapter() != this) {
            return;
        }

        //getMapControl().de
        //getMapControl().getGrid().drawGrid(mapControlDrawer);

        if (adjustedPoint != null) {
            Point2D p = null;
            if (mapAdjustedPoint != null) {
                p = mapAdjustedPoint;
            } else {
                p = getMapControl().getViewPort().toMapPoint(adjustedPoint);
            }

            if (((CADTool) cadToolStack.peek()).getVLE() == null){
                return;
            }

            if (otherMapaAdjustedPoints == null
                    || otherMapaAdjustedPoints.size() == 1) {

                ((CADTool) cadToolStack.peek())
                .drawOperation(mapControlDrawer, p.getX(), p.getY());

            }else{

                //Llamamos a la operación de dibujado con la lista de puntos
                ((CADTool) cadToolStack.peek()).drawOperation(mapControlDrawer,
                        otherMapaAdjustedPoints);
            }

        }
    }
```

Espero que le véais la utilidad y la deséis incorporar a vuestro proyecto. Sobretodo porque es una herramienta que nosotros utilizamos y de esta manera, no nos veríamos obligados a tener que recompilar vuestro proyecto para poder añadir esta herramienta ya que vendría en el paquete 'base'.

Muchas gracias. Esperamos vuestros comentarios

History

#1 - 04/18/2012 02:05 PM - Joaquín del Cerro Murciano

- Tracker changed from gvSIG bugs to gvSIG feature requests

- Status changed from New to Closed

Cierro el ticket por estar duplicado con el #541

Files

SeguimientoFinalPointSnapper.txt	8.8 KB	04/18/2012	Leticia Riestra
SeguimientoNearestPointSnapper.txt	10.2 KB	04/18/2012	Leticia Riestra
SnappersExtension.txt	5.58 KB	04/18/2012	Leticia Riestra