Application: gvSIG desktop - gvSIG feature requests #1659
Nueva Funcionalidad para gvSIG 2.0: Snapper de seguimiento

03/12/2013 03:45 PM - Leticia Riestra

Status:	Invalid	**% Done:**	0%
Priority:	Normal	**Spent time:**	0.00 hour
Assignee:			
Category:	Vector editing		
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

Estoy intentando implementar los Snappers de Seguimiento que comenté en la incidencia
https://gvsig-devel.gva.es/redmine/issues/541

Debido a que estoy intentando minimizar los cambios necesarios, he preferido abrir un hilo nuevo e intentar explicar todo paso por paso.

Mi idea es que me dijeseis cómo puedo hacer alguno de estos cambios sin tener que modificar vuestras clases (si es posible) o en el caso de que se tengan que modificar, cómo se podría hacer de forma que los cambios fuesen los menores posibles y los pudieseis aprovechar (me remito a los cambios que hicisteis en el proyecto para permitirnos mostrar nuestros formularios durante el proceso de edición).

Mis Snappers

He creado en el paquete *org.gvsig.app.project.documents.view.toolListeners.snapping.snappers* mis dos Snappers:
- SeguimientoFinalPointSnapper es el FinalPointSnapper como el vuestro pero guardando la lista de puntos de la geometría próxima sobre la que se está digitalizando.
- SeguimientoNearestPointSnapper es el NearestPointSnapper como el vuestro pero guardando la lista de puntos de la geometría próxima sobre la que se está digitalizando.

Estas dos clases usan una constante para el umbral de cercanía con el que funcionan. Lo he definido en la parte superior de la clase, pero su lugar, si lo preferís, puede ser el lugar donde definís variables constantes en gvSIG.

He modificado la clase Snapping.java del paquete *org.gvsig.app.project.documents.view.toolListeners.snapping* para añadirle el registro de estos dos nuevos Snappers (y en mi caso concreto los he puesto arriba porque quiero que tengan la máxima prioridad).

```
public static void register() {        //TODO delete when the system to load libraries is finished
        if (MapControlLocator.getMapControlManager() == null){
            MapControlLocator.registerMapControlManager(DefaultMapControlManager.class);
        }
        MapControlManager mapControlManager = MapControlLocator.getMapControlManager();

      mapControlManager.registerSnapper("SeguimientoFinalPointSnapper", SeguimientoFinalPointSnapper.class);
        mapControlManager.registerSnapper("SeguimientoNearestPointSnapper", SeguimientoNearestPointSnapper.class);
        mapControlManager.registerSnapper("FinalPointSnapper", FinalPointSnapper.class);
        mapControlManager.registerSnapper("NearestPointSnapper", NearestPointSnapper.class);
        mapControlManager.registerSnapper("PixelSnapper", PixelSnapper.class);
```

```
        mapControlManager.registerSnapper("CentralPointSnapper", CentralPointSnapper.class);
        mapControlManager.registerSnapper("QuadrantPointSnapper", QuadrantPointSnapper.class);
//        mapControlManager.registerSnapper("InsertPointSnapper", new InsertPointSnapper.class);
        mapControlManager.registerSnapper("IntersectionPointSnapper", IntersectionPointSnapper.class);
        mapControlManager.registerSnapper("MediumPointSnapper", MediumPointSnapper.class);
        mapControlManager.registerSnapper("PerpendicularPointSnapper", PerpendicularPointSnapper.class);
        mapControlManager.registerSnapper("TangentPointSnapper", TangentPointSnapper.class);
    }
```

MapControl

En esta clase necesito añadir un par de métodos
- getUsedSnap(): para devolverme el snapper que se está usando.
- setPreviousPoint: para establecer los puntos previos.

Pienso que estos dos métodos, no tienen impacto en vuestra clase MapControl y que se podrían incorporar sin problemas.

El código de dichos métodos es el siguiente:

```
    /**
     * Usado para recuperar el usedSnap que se usa en CADToolAdapter
     * @return ISnapper Snapper que se esta utilizando
     */
    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
     * @param previous
     */
    public void setPreviousPoint(double [] previous){
        previousPoint = previous;
    }
```

CADTool

En esta clase necesito añadir tres métodos
- drawOperation: para pintar la linea de puntos
- isMultiTransition, setMultiTransition que se usará debido a los snappers

El impacto de estos métodos es que debido a su definición se tienen que implementar en DefaultCADTool.

Clase CADTool

```
    /**
     * Recibe un graphics en el que se encuentra dibujada la
     * EditableFeatureSource que se pasa como parámetro. En este método, la
     * herramienta ha de implementar el dibujado de la operación que se está
     * realizando dependiendo del estado. Tendra en cuenta la lista de puntos
     * que nos pueden devolver los snappers
```

```
     *
     * @param r graphics
     * @param listaPuntos
     */
    public void drawOperation(MapControlDrawer renderer,ArrayList<Point2D> listaPuntos);

    /**
     * Comprobar si tiene que hacer multitransiciones de puntos
     * @return
     */
    public boolean isMultiTransition();
    /**
     * Establecer multitransciones de puntos al pulsar con el ratón
     * @param condicion
     */
    public void setMultiTransition(boolean condicion);
```

Clase DefaultCADTool

```
    private boolean multiTransition = false; //variable añadida para el snapping

    /* Método añadidos para el snappig */

    public void drawOperation(MapControlDrawer renderer,ArrayList<Point2D> listaPuntos){
        if(listaPuntos!=null && listaPuntos.size()>0){
            Point2D punto = (Point2D) listaPuntos.get(listaPuntos.size()-1);
            drawOperation(renderer,punto.getX(),punto.getY());
        }
    }

    //Permite transiciones múltiples para emplear los snaps de "seguir geometría"
    public boolean isMultiTransition(){
        return multiTransition;
    }

    public void setMultiTransition(boolean condicion){
        multiTransition = condicion;
    }
```

CADToolAdapter

Resulta que los Snappers que hay ahora implementados en gvSIG solo guardan las coordenadas del último punto añadido. Con mis Snappers de Seguimiento yo guardo un Array de Point2D que me permiten saber la lista de puntos que necesito. Esos puntos los calculo en mi Snapper y luego desde el CADToolAdapter lo recupero y los utilizo.

Los cambios que yo he tenido que hacer en CADToolAdapter son los siguientes:
- Cambio 1: definir los métodos para poder recuperar los puntos anteriores

```
    public void setPreviousPoint(double[] previousPoint) {
        this.previousPoint = previousPoint;
```

```
    }

    public void setPreviousPoint(Point2D punto) {
        double puntoPrevio[] = { punto.getX(), punto.getY() };
        this.previousPoint = puntoPrevio;
    }
```

- Cambio 2: declarar el arrayList que contendrá los puntos

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

- Cambio 3: 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 de seguimiento
        if (getMapControl().getUsedSnap() != null){
            if (getMapControl().getUsedSnap() instanceof SeguimientoNearestPointSnapper){
                otherMapaAdjustedPoints =
    ((SeguimientoNearestPointSnapper)getMapControl().getUsedSnap()).getSnappedPoints();
            }else if (getMapControl().getUsedSnap() instanceof SeguimientoFinalPointSnapper){
                otherMapaAdjustedPoints = ((SeguimientoFinalPointSnapper)getMapControl().getUsedSnap()).getSnappedPoints();
            }else{
                otherMapaAdjustedPoints = null;
            }
        }else{
            otherMapaAdjustedPoints = null;
        }

        showCoords(e.getPoint());

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

- Cambio 4: 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 de seguimiento
        if (getMapControl().getUsedSnap() != null){
           if (getMapControl().getUsedSnap() instanceof SeguimientoNearestPointSnapper){
              otherMapaAdjustedPoints =
    ((SeguimientoNearestPointSnapper)getMapControl().getUsedSnap()).getSnappedPoints();
           }else if (getMapControl().getUsedSnap() instanceof SeguimientoFinalPointSnapper){
              otherMapaAdjustedPoints = ((SeguimientoFinalPointSnapper)getMapControl().getUsedSnap()).getSnappedPoints();
           }else{
              otherMapaAdjustedPoints = null;
           }
        }else{
           otherMapaAdjustedPoints = null;
        }
      }
```

- Cambio 5: 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 6: redefinir el método paintComponent

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

        if (adjustedPoint != null) {
            Point2D p = null;

            p = getMapControl().getViewPort().toMapPoint(adjustedPoint);
            /*
             * We are going to draw.
             * Is mapAdjustedPoint better in any situation?
             *
            if (mapAdjustedPoint != null) {
                p = mapAdjustedPoint;
            } else {
                p = getMapControl().getViewPort().toMapPoint(adjustedPoint);
            }
            */

            // If the stack is empty, return
            if (cadToolStack.isEmpty()) {
                return;
            }

            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);
            }
```

```
        }
    }
```

Además, para dar la posibilidad al usuario de activar/desactivar el Snapper de Seguimiento de una forma directa desde la interfaz he creado una extensión que lo permite. Yo la tengo en mi plugin, pero si creeis que también puede ser útil, podemos ponerla donde queráis.
Si creéis que es interesante, os lo puedo mandar en un fichero adjunto.

Por último quedan las modificaciones que se hacen en las herramientas que basicamente es asignar el valor de previousPoint y multitransition.
En mis herramientas de digitalización que he hecho hasta ahora, las he incorporado y funciona sin problemas. Todo sería añadirlo a las vuestras (si estáis interesados claro).

Como podéis ver, para que los snappers funcionen tendría que hacer cambios en algunas de vuestras clases pero son cambios que no afectan al funcionamiento ni de los snappers que ya hay ni en las herramientas de digitalización que tenéis implementadas.
De hecho es más, si no queréis registrar el Snapper en vuestra clase Snapping porque no le veis utilidad, no pasaría nada y lo podría hacer yo directamente en mi SnapperSeguimientoExtension.

Espero haber sido más clara con este correo. Espero vuestra respuesta.

Muchas gracias y saludos

Related issues:

Related to Application: gvSIG desktop - gvSIG feature requests # 541: Añadir ...	**Invalid**	**04/18/2012**

History

#1 - 03/12/2013 05:22 PM - Leticia Riestra

En la clase CADToolAdapter también es necesario modificar el método transition para que tenga en cuenta los puntos almacenados.
La redefinición del método consiste en asignar a MapControl el valor de previousPoint cada vez que cambia.

El método quedaría de la forma siguiente:

```
    private void transition(double[] values, InputEvent event, int type) {
        questionAsked = true;
        if (!cadToolStack.isEmpty()) {
            CADTool ct = (CADTool) cadToolStack.peek();

            switch (type) {
            case ABSOLUTE:
                ct.transition(values[0], values[1], event);
                previousPoint = values;
                getMapControl().setPreviousPoint(previousPoint);
                break;
            case RELATIVE_SCU:
                // Comprobar que tenemos almacenado el punto anterior
                // y crear nuevo con coordenadas relativas a ï¿½l.
                double[] auxSCU = values;
                if (previousPoint != null) {
                    auxSCU[0] = previousPoint[0] + values[0];
```

```
            auxSCU[1] = previousPoint[1] + values[1];
        }
        ct.transition(auxSCU[0], auxSCU[1], event);

        previousPoint = auxSCU;
        getMapControl().setPreviousPoint(previousPoint);
        break;
    case RELATIVE_SCP:
        // TODO de momento no implementado.
        ct.transition(values[0], values[1], event);
        previousPoint = values;
        getMapControl().setPreviousPoint(previousPoint);
        break;
    case POLAR_SCU:// Relativo
        // Comprobar que tenemos almacenado el punto anterior
        // y crear nuevo con coordenadas relativas a ï¿½l.
        double[] auxPolarSCU = values;
        if (previousPoint != null) {
            Point2D point =
                UtilFunctions.getPoint(new Point2D.Double(
                    previousPoint[0], previousPoint[1]), Math
                    .toRadians(values[1]), values[0]);
            auxPolarSCU[0] = point.getX();
            auxPolarSCU[1] = point.getY();
            ct.transition(auxPolarSCU[0], auxPolarSCU[1], event);
        } else {
            Point2D point =
                UtilFunctions.getPoint(new Point2D.Double(0, 0),
                    Math.toRadians(values[1]), values[0]);
            auxPolarSCU[0] = point.getX();
            auxPolarSCU[1] = point.getY();
            ct.transition(auxPolarSCU[0], auxPolarSCU[1], event);
        }
        previousPoint = auxPolarSCU;
        getMapControl().setPreviousPoint(previousPoint);
        break;
    case POLAR_SCP:// Absoluto
        double[] auxPolarSCP = values;
        if (previousPoint != null) {
            Point2D point =
                UtilFunctions.getPoint(new Point2D.Double(0, 0),
                    Math.toRadians(values[1]), values[0]);
            auxPolarSCP[0] = point.getX();
            auxPolarSCP[1] = point.getY();
            ct.transition(auxPolarSCP[0], auxPolarSCP[1], event);
        } else {
            Point2D point =
                UtilFunctions.getPoint(new Point2D.Double(0, 0),
                    values[1], values[0]);
            auxPolarSCP[0] = point.getX();
            auxPolarSCP[1] = point.getY();
            ct.transition(auxPolarSCP[0], auxPolarSCP[1], event);
        }
```

```
                previousPoint = auxPolarSCP;
                getMapControl().setPreviousPoint(previousPoint);
                break;
            default:
                break;
            }
            askQuestion();
        }
        configureMenu();
        PluginServices.getMainFrame().enableControls();
    }
```

#2 - 01/15/2014 12:19 PM - Álvaro Anguix

- Category set to Vector editing

#3 - 01/07/2015 01:51 PM - Álvaro Anguix

- Status changed from New to Invalid

Paso a invalid ya que más que una fr eran una serie de consultas sobre una funcionalidad, que por otro lado, desconocemos y no es pública.