```
/**
 * ------------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ------------------------------------------------------------------------------
 */

package dk.ruc.blaster.export;

import java.awt.Dimension;
import java.io.File;
import java.io.IOException;
import java.util.Iterator;

import javax.swing.JOptionPane;

import dk.ruc.blaster.model.Map;
import dk.ruc.blaster.model.Triangle;
import dk.ruc.blaster.model.Vector;
import dk.ruc.blaster.model.Vertex;
import dk.ruc.blaster.ui.BlasterEditor;
import electric.xml.Document;
import electric.xml.Element;

/**
 * Static class to export a Map to a xml file
 *
 * Last change by: $Author: dyregod $
 *
 * $Header: /var/cvs/Alwazah\040Editor/source/dk/ruc/blaster/export/Exporter.jav
a,v 1.15 2002/12/16 00:19:08 dyregod Exp $
 *
 * @version $Revision: 1.15 $
 */
public class Exporter
{
    /**
     * Export method. Does the actual map -> xml conversion
     * @param map the map file to export
     * @param file the xml file to be written to
     */
    public static void export(Map map, File file)
    {

        // create empty document
        Document document = new Document();

        // create root
        Element mapElement = document.setRoot("map");
        mapElement.setAttribute("width", String.valueOf(map.width));
        mapElement.setAttribute("height", String.valueOf(map.height));
        mapElement.setAttribute("gravityx", String.valueOf(map.gravityx));
        mapElement.setAttribute("gravityy", String.valueOf(map.gravityy));
        mapElement.setAttribute("name", map.name);

        // add points
        Element points = mapElement.addElement("points");
        points.setAttribute("size", String.valueOf(map.vertices.size()));

        Dimension d = map.getDimensions();
```



```
        for (Iterator iter = map.vertices.iterator(); iter.hasNext();)
        {
            Vertex v = (Vertex) iter.next();
            Vector p = v.getPosition();

            Element point = points.addElement("point");
            point.setAttribute("x", String.valueOf(p.x - d.width / 2));
            point.setAttribute("y", String.valueOf(p.y - d.height / 2));
            point.setAttribute("id", String.valueOf(map.vertices.indexOf(v)));
        }

        Element polygons = mapElement.addElement("polygons");
        polygons.setAttribute("size", String.valueOf(map.triangles.size()));

        for (Iterator iter = map.triangles.iterator(); iter.hasNext();)
        {
            Triangle triangle = (Triangle) iter.next();

            Element polygon = polygons.addElement("polygon");
            polygon.setAttribute(
                "id",
                String.valueOf(map.triangles.indexOf(triangle)));

            // add color element
            Element color = polygon.addElement("color");
            color.setAttribute(
                "r",
                String.valueOf(triangle.getColor().getRed() / 255.0f));
            color.setAttribute(
                "g",
                String.valueOf(triangle.getColor().getGreen() / 255.0f));
            color.setAttribute(
                "b",
                String.valueOf(triangle.getColor().getBlue() / 255.0f));

            Element edgeElement;
            //  Edge edge;
            // add (hopefully) 3 edges
            int[] v = new int[3];
            Vertex ver = triangle.vertices[2];
            Vector vv = ver.getPosition();
            float dot =
                vv.subtract(triangle.edges[0].getB().getPosition()).dot(
                    triangle.edges[0].normal());
            if (dot < 0)
            {
                v[0] = map.vertices.indexOf(triangle.edges[0].getA());
                v[1] = map.vertices.indexOf(triangle.edges[0].getB());
                v[2] = map.vertices.indexOf(ver);
                edgeElement = polygon.addElement("edge");
                edgeElement.setAttribute("p1", String.valueOf(v[0]));
                edgeElement.setAttribute(
                    "platform",
                    triangle.edges[0].getEdgeType() ? "true" : "false");
                edgeElement.setAttribute("id", "0");
                edgeElement = polygon.addElement("edge");
                edgeElement.setAttribute("p1", String.valueOf(v[1]));
                edgeElement.setAttribute(
                    "platform",
                    triangle.edges[1].getEdgeType() ? "true" : "false");
                edgeElement.setAttribute("id", "1");
                edgeElement = polygon.addElement("edge");
```

```
                edgeElement.setAttribute("p1", String.valueOf(v[2]));
                edgeElement.setAttribute(
                    "platform",
                    triangle.edges[2].getEdgeType() ? "true" : "false");
                edgeElement.setAttribute("id", "2");
            }
            else
            {
                v[0] = map.vertices.indexOf(triangle.edges[0].getB());
                v[1] = map.vertices.indexOf(ver);
                v[2] = map.vertices.indexOf(triangle.edges[0].getA());
                edgeElement = polygon.addElement("edge");
                edgeElement.setAttribute("p1", String.valueOf(v[2]));
                edgeElement.setAttribute(
                    "platform",
                    triangle.edges[0].getEdgeType() ? "true" : "false");
                edgeElement.setAttribute("id", "0");
                edgeElement = polygon.addElement("edge");
                edgeElement.setAttribute("p1", String.valueOf(v[1]));
                edgeElement.setAttribute(
                    "platform",
                    triangle.edges[1].getEdgeType() ? "true" : "false");
                edgeElement.setAttribute("id", "1");
                edgeElement = polygon.addElement("edge");
                edgeElement.setAttribute("p1", String.valueOf(v[0]));
                edgeElement.setAttribute(
                    "platform",
                    triangle.edges[2].getEdgeType() ? "true" : "false");
                edgeElement.setAttribute("id", "2");
            }

        }

        try
        {
            document.write(file);
            if (BlasterEditor.currentFrame == null)
                return;
            else
                JOptionPane.showInternalMessageDialog(
                    BlasterEditor.currentFrame.getContentPane(),
                    "Map written to file: " + file.getPath(),
                    "File saved",
                    JOptionPane.INFORMATION_MESSAGE);
        }
        catch (IOException e)
        {
            if (BlasterEditor.currentFrame == null)
                return;
            else
                JOptionPane.showInternalMessageDialog(
                    BlasterEditor.currentFrame.getContentPane(),
                    "Could not write file due to:\n" + e.getMessage(),
                    "Error",
                    JOptionPane.ERROR_MESSAGE);
        }

    }

}
```



```
/**
 * ----------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ----------------------------------------------------------------------------
 */

package dk.ruc.blaster.model;

import java.awt.Point;
import java.awt.Rectangle;
import java.util.HashMap;
/**
 * Represents an edge
 *
 * Last change by: $Author: dyregod $
 *
 * $Header: /var/cvs/Alwazah\040Editor/source/dk/ruc/blaster/model/Edge.java,v 1
.9 2002/12/16 00:19:08 dyregod Exp $
 *
 * @version $Revision: 1.9 $
 * @author dyregod
 */
public class Edge extends EditorObject
{
    Vertex a;
    Vertex b;
    private boolean isPlatform = false;

    /**
     * Constructor Edge.
     * @param vertex
     * @param vertex1
     */
    public Edge(Vertex a, Vertex b)
    {
        this.a = a;
        this.b = b;

    }

    /**
     * Returns the a vertex
     * @return Vertex
     */
    public Vertex getA()
    {
        return a;
    }

    public void setA( Vertex v )
    {
        a = v;
    }

    /**
     * Returns the b vertex
     * @return Vertex
     */
    public Vertex getB()
```

```
    {
        return b;
    }
    /**
     * Returns the normal to the edge
     * @return Vector
     */
    public Vector normal()
    {
        return b.position.subtract(a.position).normalize().ortho();
    }

    /**
     * @see dk.ruc.blaster.model.EditorObject#bounds()
     */
    public Rectangle bounds()
    {
        float ax = a.position.x;
        float ay = a.position.y;
        float bx = b.position.x;
        float by = b.position.y;

        if (ax > bx)
        {
            float temp = ax;
            ax = bx;
            bx = temp;
        }
        if (ay > by)
        {
            float temp = ay;
            ay = by;
            by = temp;
        }

        final int x = (int)(ax+0.5f);
        final int y = (int)(ay+0.5f);
        final int width = (int)((bx - ax)+0.5f);
        final int height = (int)((by - ay)+0.5f);

        return new Rectangle( x, y, width, height );
    }

    /**
     * Finds the distance from the edge to a point p
     * @param p
     * @return float
     */
    public float getDistanceToPoint( Point p )
    {
        if ( !bounds().contains( p ) )
        {
            return Float.MAX_VALUE;
        }

        final float x0 = (float) p.getX();
        final float y0 = (float) p.getY();

        final float x1 = a.position.x;
        final float y1 = a.position.y;
        final float x2 = b.position.x;
        final float y2 = b.position.y;
```



```
        // Calculate the distance between the point and the line.
        float dist = ( float ) (( Math.abs( (y2 - y1) * (x0 - x1) - (x2 - x1) *
(y0 - y1) ) )/ Math.sqrt( Math.pow( (x2 - x1), 2 ) + Math.pow( (y2 - y1), 2 ) ))
;

        return dist;
    }

    /**
     * @see dk.ruc.blaster.model.EditorObject#setProperties(java.util.HashMap)
     */
    public void setProperties(HashMap list)
    {
        String s = (String) list.get("platform");
        if ( s.equals("true") )
        {
            isPlatform = true;
        }
        else
        {
            isPlatform = false;
        }
    }

    /**
     * @see dk.ruc.blaster.model.EditorObject#getProperties()
     */
    public HashMap getProperties()
    {
        HashMap map = new HashMap();
        map.put("platform", isPlatform ? "true" : "false" );

        return map;
    }

    protected boolean selected = false;

    public void select( boolean value )
    {
        System.out.println("select " + value);
        selected = value;
    }

    /**
     * Returns <code>true</code> if edge is a platform
     * @return boolean
     */
    public boolean getEdgeType ()
    {
        return isPlatform;
    }

    public void move (int dx, int dy)
    {
        a.move( dx, dy );
        b.move( dx, dy );
    }

    public Rectangle updateOnDrag()
    {
        final Rectangle r = new Rectangle();

        r.add(a.updateOnDrag());
```

```
        r.add(b.updateOnDrag());

        return r;
    }
}
```



```
/**
 * ------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ------------------------------------------------------------------------
 */

package dk.ruc.blaster.model;

import java.awt.Rectangle;
import java.io.Serializable;
import java.util.HashMap;
/**
 * The super class of all objects in the editor
 *
 * Last change by: $Author: dyregod $
 *
 * $Header: /var/cvs/Alwazah\040Editor/source/dk/ruc/blaster/model/EditorObject.
java,v 1.5 2002/12/16 00:19:08 dyregod Exp $
 *
 * @version $Revision: 1.5 $
 * @author dyregod
 */
public abstract class EditorObject implements Serializable
{
    /**
     * Returns the objects bounding box
     * @return Rectangle
     */
    public abstract Rectangle bounds();
    /**
     * Sets this objects properties
     * @param list
     */
    public abstract void setProperties(HashMap list);
    /**
     * Get properties for this obejct
     * @return HashMap
     */
    public abstract HashMap getProperties();
    /**
     *
     */
    protected boolean selected = false;
    /**
     * Move object by dx, dy
     * @param dx
     * @param dy
     */
    public abstract void move (int dx, int dy);

    public abstract Rectangle updateOnDrag();

    public void select( boolean value )
    {
        System.out.println("select " + value);
        selected = value;
    }

}
```

```
/**
 * ------------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ------------------------------------------------------------------------------
 */

package dk.ruc.blaster.model;

import java.awt.Dimension;
import java.awt.Point;
import java.awt.Rectangle;
import java.io.Serializable;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

import dk.ruc.blaster.ui.BlasterEditor;

public class Map extends EditorObject implements Serializable
{
    // list of stripes
    // list of fans
    // list of colors?
    // list of polygons
    public List vertices;
    public List triangles;

    public float gravityx = 0;
    public float gravityy = 0;
    public float width = 0;
    public float height = 0;
    public String name = "";

    private Dimension size;

    public Map()
    {
        width = 1000;
        height = 1000;
        vertices = new LinkedList();
        triangles = new LinkedList();
    }

    public Dimension getDimensions()
    {
        return new Dimension((int) width, (int) height);
    }
    public Rectangle bounds()
    {
        return new Rectangle((int) width, (int) height);
    }

    List getVerticesAtPoint(Point p)
    {
        List hit = new LinkedList();

        Vertex v;
```



```
        Iterator i = vertices.iterator();

        while (i.hasNext())
        {
            v = (Vertex) i.next();

            if (v.contains(p))
            {
                hit.add(v);
            }
        }

        return hit;
    }

    public List getVerticesInRect(Rectangle bounds)
    {
        // At some point in the future, replace this with some sort of tree.
        // For now, just do a linear search and compare with bounds

        List visible = new LinkedList();

        Vertex v;

        Iterator i = vertices.iterator();

        while (i.hasNext())
        {
            v = (Vertex) i.next();

            if (bounds.intersects(v.bounds()))
            {
                visible.add(v);
            }
        }

        return visible;
    }

    Edge getNearestEdgeAtPoint(Point p)
    {
        float maxDist = Float.MAX_VALUE;

        Edge edge = null;

        Triangle t;

        Iterator i = triangles.iterator();

        while (i.hasNext())
        {
            t = (Triangle) i.next();

            for (int i2 = 0; i2 < t.edges.length; i2++)
            {
                float temp =
                    (float) Math.abs(t.edges[i2].getDistanceToPoint(p));
System.err.println("dist "+i2+" "+temp);
                if (temp < maxDist && temp <= 3.0f)
                {
                    maxDist = temp;
                    edge = t.edges[i2];
                }
```

```
            }
        }
        return edge;
    }

    List getTrianglesAtPoint(Point p)
    {
        LinkedList l = new LinkedList();

        Triangle t;

        Iterator i = triangles.iterator();

        while (i.hasNext())
        {
            t = (Triangle) i.next();

            if (t.contains(p))
            {
                l.add(t);
            }
        }

        return l;
    }

    public List getTrianglesInRect(Rectangle bounds)
    {
        // This needs to be improved

        LinkedList visible = new LinkedList();

        Triangle t;

        Iterator i = triangles.iterator();

        while (i.hasNext())
        {
            t = (Triangle) i.next();

            if (bounds.intersects(t.bounds()))
            {
                visible.add(t);
            }
        }

        return visible;
    }

    public EditorObject select(Point p)
    {
        EditorObject obj;
        obj = getNearestVertexToPoint(p, null);
        if (obj != null)
            return obj;

        obj = getNearestEdgeAtPoint(p);
        if (obj != null)
        {
            return obj;
        }

        List list = getTrianglesAtPoint(p);
```



```
        if (list.size() > 0)
        {
            obj = (EditorObject) list.get(0);
            return obj;
        }
        return this;
    }

    public Vertex getNearestVertexToPoint(Point p, Vertex ignore)
    {
        final Vertex temp = new Vertex(p.x, p.y);

        List vectors = getVerticesAtPoint(p);

        if (vectors.isEmpty())
        {
            return null;
        }

        Vertex v;
        Vertex nearestVertice = null;
        float nearestLength = Float.MAX_VALUE;
        float length;

        final Iterator i = vectors.iterator();

        while (i.hasNext())
        {
            v = (Vertex) i.next();

            length = temp.squaredDistanceTo(v);

            if (length < nearestLength && v != ignore)
            {
                nearestLength = length;
                nearestVertice = v;
            }
        }

        return nearestVertice;
    }

    public Vertex addVertex(Point p)
    {
        Vertex v = getNearestVertexToPoint(p, null);

        if (v == null)
        {
            v = new Vertex(p.x, p.y);

            // We didn't find any preexisting vertices, so it's ok to add the ne
w vertice to the map
            vertices.add(v);
        }

        return v;
    }

    public Triangle addTriangle(Vertex verts[])
    {
        final Triangle t = new Triangle(verts);

        triangles.add(t);
```

Map.java


```
        return t;
    }
    /**
     * @see dk.ruc.blaster.model.EditorObject#-(HashMap)
     */
    public void setProperties(HashMap list)
    {
        name = (String) list.get("name");
        gravityx = Float.valueOf((String) list.get("gravityx")).floatValue();
        gravityy = Float.valueOf((String) list.get("gravityy")).floatValue();
        width = Float.valueOf((String) list.get("width")).floatValue();
        height = Float.valueOf((String) list.get("height")).floatValue();
        if (BlasterEditor.currentFrame != null)
        {
            BlasterEditor.currentFrame.getCanvas().setSize(
                new Dimension((int) width, (int) height));
            BlasterEditor.currentFrame.setTitle(name);
        }
    }
    /**
     * @see dk.ruc.blaster.model.EditorObject#getProperties()
     */
    public HashMap getProperties()
    {
        HashMap map = new HashMap();
        map.put("width", String.valueOf(width));
        map.put("height", String.valueOf(height));
        map.put("gravityx", String.valueOf(gravityx));
        map.put("gravityy", String.valueOf(gravityy));
        map.put("name", name);
        return map;
    }
    public void unselectall()
    {
        for (Iterator iter = vertices.iterator(); iter.hasNext();)
        {
            EditorObject element = (EditorObject) iter.next();
            element.select(false);
        }
        for (Iterator iter = triangles.iterator(); iter.hasNext();)
        {
            EditorObject element = (EditorObject) iter.next();
            element.select(false);
        }
    }

    public void move(int dx, int dy)
    {
    }

    public Rectangle updateOnDrag()
    {
        return null;
    }
}
```

Triangle.java


```
/**
 * ----------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ----------------------------------------------------------------------------
 */

package dk.ruc.blaster.model;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Point;
import java.awt.Polygon;
import java.awt.Rectangle;
import java.io.Serializable;
import java.util.HashMap;

public class Triangle extends EditorObject implements Serializable
{
    private final Color selectedColor = new Color( 0.8f, 1.0f, 0.8f );
    private final Color fillColor = Color.white;
    private final Color borderColor = Color.black;

    private Color color = Color.white;

    public Vertex vertices[];
    public Edge edges[];
    private Polygon poly = null;
    private Rectangle bounds = null;


    public Triangle( Vertex verts[] )
    {
        vertices = verts;
        edges = new Edge[ verts.length ];

        for ( int i = 0; i < verts.length; i++ )
        {
            verts[i].addPoly( this );
            edges[i] = new Edge( verts[i], verts[(i+1)%verts.length] );
        }

        update();
    }

    public void update()
    {
        // reorder verts to maintain clockwise orientation?

        final int xcoords[] = new int[4];
        final int ycoords[] = new int[4];

        for ( int i = 0; i < 3; i++ )
        {
            xcoords[i] = (int)vertices[i].position.x;
            ycoords[i] = (int)vertices[i].position.y;
        }

        xcoords[3] = xcoords[0];
        ycoords[3] = ycoords[0];
```

```
        poly = new Polygon( xcoords, ycoords, 4 );
        bounds = poly.getBounds();
    }

    public boolean contains( Point p )
    {
        return poly.contains( p );
    }

    public Rectangle bounds()
    {
        return bounds;
    }

    public void draw( Graphics g )
    {
        final Color c = g.getColor();

        g.setColor( selected ? selectedColor : color );
        g.fillPolygon( poly );

        g.setColor( borderColor );
        g.drawPolygon( poly );

        g.setColor( c );
    }
    /**
     * @see dk.ruc.blaster.model.EditorObject#setProperties(HashMap)
     */
    public void setProperties(HashMap list)
    {
        float r = Float.valueOf((String)list.get("r")).floatValue();
        System.out.println(r);
        float g = Float.valueOf((String)list.get("g")).floatValue();
        float b = Float.valueOf((String)list.get("b")).floatValue();

        setColor(new Color(r,g,b));
    }

    /**
     * Returns the color.
     * @return Color
     */
    public Color getColor()
    {
        return color;
    }

    /**
     * Sets the color.
     * @param color The color to set
     */
    public void setColor(Color color)
    {
        this.color = color;
    }

    /**
     * @see dk.ruc.blaster.model.EditorObject#getProperties()
     */
    public HashMap getProperties()
    {
```



```
        HashMap map = new HashMap();
        float[] comp = new float[3];
        color.getRGBColorComponents(comp);
        map.put("r", String.valueOf(comp[0]));
        map.put("g", String.valueOf(comp[1]));
        map.put("b", String.valueOf(comp[2]));
        return map;
    }

    public void move (int dx, int dy)
    {
        for ( int i = 0; i < vertices.length; i++ )
        {
            vertices[i].move( dx, dy );
        }
    }

    public Rectangle updateOnDrag()
    {
        final Rectangle r = new Rectangle();

        for( int i = 0; i < vertices.length; i++ )
        {
            r.add( vertices[i].updateOnDrag() );
        }

        return r;
    }

}
```

Vector.java


```
/**
 * ------------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ------------------------------------------------------------------------------
 */

package dk.ruc.blaster.model;

import java.io.Serializable;

public class Vector implements Serializable
{
    public float x;
    public float y;

    public Vector( float X, float Y )
    {
        x = X;
        y = Y;
    }

    public Vector add( Vector v )
    {
        return new Vector( this.x + v.x, this.y + v.y );
    }

    public Vector subtract( Vector v )
    {
        return new Vector( this.x - v.x, this.y - v.y );
    }

    public float length2()
    {
        return x*x + y*y;
    }

    public float length()
    {
        return (float)Math.sqrt( length2() );
    }
    public Vector scale(float scale)
    {
        return new Vector(x*scale, y*scale);
    }
    public Vector normalize()
    {
        return this.scale(1/length());
    }
    public Vector ortho()
    {
        return new Vector(-y,x);
    }
    public float dot(Vector b)
    {
        return x*b.x + y*b.y;
    }

}
```

Vertex.java


```
/**
 * ------------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ------------------------------------------------------------------------------
 */

package dk.ruc.blaster.model;

import java.awt.Graphics;
import java.awt.Point;
import java.awt.Rectangle;
import java.io.Serializable;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

public class Vertex extends EditorObject implements Serializable
{
    Vector position;

    private boolean selected = false;

    private List triangles = new LinkedList();

    final public static int THRESHHOLD = 3;


    public Vertex( float a, float b )
    {
        position = new Vector( a, b );
    }

    public void select( boolean value )
    {
        selected = value;
    }

    public void move( int dx, int dy )
    {
        position.x += dx;
        position.y += dy;
    }

    public void draw( Graphics g )
    {
        final int ix = (int)(position.x+0.5f);
        final int iy = (int)(position.y+0.5f);

        g.drawRect( ix-1, iy-1, 2, 2 );
    }

    public void addPoly( Triangle t )
    {
        triangles.add( t );
    }

    public void removePoly(Triangle t)
    {
```

```
        triangles.remove(t);
    }

    public Rectangle updateOnDrag()
    {
        Triangle t;

        final Rectangle r = new Rectangle();

        Iterator i = triangles.iterator();

        while ( i.hasNext() )
        {
            t = (Triangle)i.next();

            r.add( t.bounds() );
            t.update();
        }

        return r;
    }

    public Rectangle bounds()
    {
        final int ix = (int)(position.x+0.5f);
        final int iy = (int)(position.y+0.5f);

        return new Rectangle( ix-THRESHHOLD, iy-THRESHHOLD, THRESHHOLD*2, THRESH
HOLD*2 );
    }

    public boolean contains( Point p )
    {
        return bounds().contains( p );
    }

    public float squaredDistanceTo( Vertex v )
    {
        final Vector vec = position.subtract( v.position );

        return vec.length2();
    }
    /**
     * @see dk.ruc.blaster.model.EditorObject#setProperties(HashMap)
     */
    public void setProperties(HashMap list)
    {
    }

    /**
     * @see dk.ruc.blaster.model.EditorObject#getProperties()
     */
    public HashMap getProperties()
    {
        return null;
    }

    /**
    * Returns the position.
    * @return Vector
    */
    public Vector getPosition()
    {
```



```
        return position;
    }

    public int getNumberOfAttachedPolygons()
    {
        return triangles.size();
    }
}
```

```
/**
 * ----------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ----------------------------------------------------------------------------
 */

package dk.ruc.blaster.ui;

import java.awt.Rectangle;
import java.awt.event.KeyEvent;
import java.awt.event.MouseEvent;

import dk.ruc.blaster.model.*;

abstract public class AbstractMode
{
    protected Map map;

    public AbstractMode( Map m )
    {
        map = m;
    }

    public void activate()
    {
    }

    public void deactivate()
    {
    }

    public Rectangle keyPressed( KeyEvent e )
    {
        return null;
    }

    public Rectangle keyReleased( KeyEvent e )
    {
        return null;
    }

    public Rectangle keyTyped( KeyEvent e )
    {
        return null;
    }

    public Rectangle mousePressed( MouseEvent e )
    {
        return null;
    }

    public Rectangle mouseReleased( MouseEvent e )
    {
        return null;
    }

    public Rectangle mouseClicked( MouseEvent e )
    {
        return null;
```



```
    }

    public Rectangle mouseDragged( MouseEvent e )
    {
        return null;
    }

    public Rectangle mouseMoved( MouseEvent e )
    {
        return null;
    }

    public Rectangle mouseEntered( MouseEvent e )
    {
        return null;
    }

    public Rectangle mouseExited( Rectangle e )
    {
        return null;
    }
}
```

```
/**
 * ------------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ------------------------------------------------------------------------------
 */

package dk.ruc.blaster.ui;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ComponentEvent;
import java.awt.event.ComponentListener;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyVetoException;
import java.beans.VetoableChangeListener;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JDesktopPane;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JToolBar;

import dk.ruc.blaster.export.Exporter;
import dk.ruc.blaster.model.Map;
import dk.ruc.blaster.ui.property.PropertyFloat;

public class BlasterEditor
    extends JFrame
    implements VetoableChangeListener, ComponentListener
{
    JDesktopPane desktop = new JDesktopPane();
    MapFrame mapFrame;

    public static MapFrame currentFrame;
    public static int windowCount = 0;

    static void switchMode()
    {
        if (currentFrame == null)
            return;
        currentFrame.switchMode();

    }
```



```
    public void buildUI()
    {
        this.setTitle("BlasterEditor - The One and Only");

        JPanel panel1 = new JPanel();
        panel1.setLayout(new BorderLayout());
        JPanel panel2 = new JPanel();
        panel2.setLayout(new BorderLayout());
        panel1.add(createMenuBar(), BorderLayout.NORTH);
        panel2.add(createToolBar(), BorderLayout.CENTER);
        panel1.add(panel2, BorderLayout.SOUTH);

        desktop.add(FloatingToolBar.getInstance());
        desktop.add(PropertyFloat.getInstance());

        desktop.setBorder(BorderFactory.createEtchedBorder());
        FloatingToolBar.getInstance().setSize(60, 60);
        FloatingToolBar.getInstance().setVisible(true);
        FloatingToolBar.getInstance().setLocation(20, 60);

        this.getContentPane().add(panel1, BorderLayout.NORTH);
        this.getContentPane().add(desktop, BorderLayout.CENTER);

        this.setSize(640, 480);
        Dimension size = Toolkit.getDefaultToolkit().getScreenSize();
        this.setLocation(
            (size.width - this.getSize().width) / 2,
            (size.height - this.getSize().height) / 2);
    }

    public JMenuBar createMenuBar()
    {
        JMenuBar menubar = new JMenuBar();
        JMenu filemenu = new JMenu();
        filemenu.setText("File");
        JMenuItem itemNew = new JMenuItem();
        itemNew.setText("New");
        itemNew.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                newFrame();
            }
        });
        JMenuItem itemOpen = new JMenuItem();
        itemOpen.setText("Open");
        itemOpen.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent ae)
            {
                load();
            }
        });

        JMenuItem itemSave = new JMenuItem();
        itemSave.setText("Save");
        itemSave.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                save();
            }
```

```
        });

        JMenuItem itemExport = new JMenuItem();
        itemExport.setText("Export");
        itemExport.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                export();
            }
        });

        JMenuItem itemExit = new JMenuItem();
        itemExit.setText("Exit");

        filemenu.add(itemNew);
        filemenu.addSeparator();
        filemenu.add(itemOpen);
        filemenu.add(itemSave);
        filemenu.addSeparator();
        filemenu.add(itemExport);
        filemenu.addSeparator();
        filemenu.add(itemExit);

        menubar.add(filemenu);

        return menubar;
    }
    public JToolBar createToolBar()
    {
        JToolBar toolbar = new JToolBar();
        JButton buttonNew = new JButton("New");
        JButton buttonOpen = new JButton("Open");
        JButton buttonSave = new JButton("Save");
        JButton buttonExport = new JButton("Export");
        buttonNew.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                newFrame();
            }
        });
        buttonOpen.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                load();
            }
        });
        buttonSave.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                save();
            }
        });
        buttonExport.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                export();
            }
        });
```



```
        toolbar.add(buttonNew);
        toolbar.add(buttonOpen);
        toolbar.add(buttonSave);
        toolbar.add(buttonExport);

        return toolbar;
    }
    private void load()
    {
        try
        {
            boolean test = false;
            if (currentFrame != null)
            {
                test = currentFrame.getLastUsedDir() == null;
            }
            else
                test = true;
            JFileChooser filechooser =
                new JFileChooser(
                    test
                        ? new File(System.getProperty("user.dir"))
                        : currentFrame.getLastUsedDir());
            filechooser.setFileFilter(
                new FileFilter(".bin", "Editor binaries"));
            filechooser.showOpenDialog(mapFrame);
            File file = filechooser.getSelectedFile();
            if (file == null)
                return;
            FileInputStream istream = new FileInputStream(file);
            ObjectInputStream p = new ObjectInputStream(istream);

            Map map = (Map) p.readObject();

            istream.close();

            MapFrame frame = new MapFrame(map);
            desktop.add(frame, new Integer(-1));
            frame.setSize(400, 300);
            frame.display();
        }
        catch (FileNotFoundException e)
        {
            e.printStackTrace();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
        catch (ClassNotFoundException e)
        {
            e.printStackTrace();
        }
    }
    private void export()
    {
        boolean test = false;
        if (currentFrame != null)
        {
            test = currentFrame.getLastUsedDir() == null;
        }
        else
            test = true;
```

```
        JFileChooser filechooser =
            new JFileChooser(
                test
                    ? new File(System.getProperty("user.dir"))
                    : currentFrame.getLastUsedDir());
        filechooser.setFileFilter(new FileFilter(".map", "Blaster Map files"));
        filechooser.showSaveDialog(mapFrame);
        File file = filechooser.getSelectedFile();
        if (file == null)
            return;
        currentFrame.setLastUsedDir(file.getParentFile());
        Exporter.export(currentFrame.map, file);
    }
    private void save()
    {
        try
        {

            boolean test = false;
            if (currentFrame != null)
            {
                test = currentFrame.getLastUsedDir() == null;
            }
            else
                test = true;
            JFileChooser filechooser =
                new JFileChooser(
                    test
                        ? new File(System.getProperty("user.dir"))
                        : currentFrame.getLastUsedDir());
            filechooser.setFileFilter(
                new FileFilter(".bin", "Editor binaries"));
            filechooser.showSaveDialog(mapFrame);
            File file = filechooser.getSelectedFile();
            if (file == null)
                return;
            currentFrame.setLastUsedDir(file.getParentFile());
            FileOutputStream ostream = new FileOutputStream(file);
            ObjectOutputStream p = new ObjectOutputStream(ostream);

            p.writeObject(currentFrame.map);

            p.flush();
            ostream.close();
            JOptionPane.showInternalMessageDialog(
                BlasterEditor.currentFrame.getContentPane(),
                "Map written to file: " + file.getPath(),
                "File saved",
                JOptionPane.INFORMATION_MESSAGE);
        }
        catch (Exception ee)
        {
            JOptionPane.showInternalMessageDialog(
                BlasterEditor.currentFrame.getContentPane(),
                "Could not write file due to:\n" + ee.getMessage(),
                "Error",
                JOptionPane.ERROR_MESSAGE);
        }
    }
    private void newFrame()
    {
        MapFrame frame = new MapFrame("Untitled " + windowCount);
        desktop.add(frame, new Integer(-1));
```



```
        frame.setSize(400, 300);
        frame.display();
    }
    public static void main(String[] args)
    {
        BlasterEditor editor = new BlasterEditor();
        /*      editor.addWindowListener(new WindowAdapter()
                {
                    public void windowClosing(WindowEvent e)
                    {
                        System.exit(0);
                    }
                });*/
        editor.buildUI();
        editor.setVisible(true);
    }
    /**
     * @see java.awt.event.ComponentListener#componentHidden(ComponentEvent)
     */
    public void componentHidden(ComponentEvent e)
    {
    }

    /**
     * @see java.awt.event.ComponentListener#componentMoved(ComponentEvent)
     */
    public void componentMoved(ComponentEvent e)
    {
    }

    /**
     * @see java.awt.event.ComponentListener#componentResized(ComponentEvent)
     */
    public void componentResized(ComponentEvent e)
    {
    }

    /**
     * @see java.awt.event.ComponentListener#componentShown(ComponentEvent)
     */
    public void componentShown(ComponentEvent e)
    {
    }

    /**
     * @see java.beans.VetoableChangeListener#vetoableChange(PropertyChangeEvent
)
     */
    public void vetoableChange(PropertyChangeEvent evt)
        throws PropertyVetoException
    {

    }

}
class FileFilter extends javax.swing.filechooser.FileFilter
{
    private String suffix;
    private String description;

    public FileFilter(String suffix, String description)
    {
        this.suffix = suffix;
```

```
        this.description = description;
    }
    public boolean accept(File f)
    {
        if (f.isDirectory())
            return true;
        else
            return f.getPath().endsWith(suffix);
    }

    public String getDescription()
    {
        return description;
    }
}
```



```
/**
 * ----------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ----------------------------------------------------------------------------
 */

package dk.ruc.blaster.ui;

import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.util.Iterator;
import java.util.List;

import javax.swing.JPanel;

import dk.ruc.blaster.model.Map;
import dk.ruc.blaster.model.Triangle;
import dk.ruc.blaster.model.Vertex;

public class EditorCanvas extends JPanel
{
    final private Map map;

    public EditorCanvas(Map map)
    {
        this.map = map;
    }

    public Dimension getPreferredSize()
    {
        return map.getDimensions();
    }

    public Dimension getMinimumSize()
    {
        return map.getDimensions();
    }

    public Dimension getMaximumSize()
    {
        return map.getDimensions();
    }

    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);

        // Get the part of the canvas that needs to be redrawn
        Rectangle bounds = g.getClipBounds();

        // Draw the axes
        Dimension d = map.getDimensions();
        g.drawLine( d.width/2, 0, d.width/2, d.height );
        g.drawLine( 0, d.height/2, d.width, d.height/2 );

        // Repaint the polygons that lie within the bounds
        List verts = map.getVerticesInRect(bounds);
        List tris = map.getTrianglesInRect(bounds);
```

```
        Iterator i = null;

        Triangle t = null;

        i = tris.iterator();

        while (i.hasNext())
        {
            t = (Triangle) i.next();

            t.draw(g);
        }

        Vertex v = null;

        i = verts.iterator();

        while (i.hasNext())
        {
            v = (Vertex) i.next();

            v.draw(g);
        }
    }
}
```



```
/**
 * ----------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ----------------------------------------------------------------------------
 */

package dk.ruc.blaster.ui;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.beans.PropertyVetoException;

import javax.swing.ButtonGroup;
import javax.swing.JInternalFrame;
import javax.swing.JToggleButton;

/*
 * Floating toolbar class.
 *
 * Last change by: $Author$
 *
 * $Header$
 *
 * @version $Revision$
 * @author dk13043
 */
public class FloatingToolBar extends JInternalFrame
{
    private static FloatingToolBar instance = null;

    private JToggleButton polygonModeButton;
    private JToggleButton moveModeButton;

    public static FloatingToolBar getInstance()
    {
        if (instance == null)
        {
            instance = new FloatingToolBar();
        }
        return instance;
    }
    private FloatingToolBar()
    {
        getContentPane().setLayout(new GridLayout(2, 2));

        ButtonGroup group = new ButtonGroup();

        polygonModeButton = new JToggleButton("P");
        polygonModeButton.addActionListener(new ActionListener()
        {

            public void actionPerformed(ActionEvent e)
            {
                BlasterEditor.currentFrame.getCurrentMode().deactivate();
                BlasterEditor.currentFrame.setCurrentMode(BlasterEditor.currentF
rame.polygonMode);
                BlasterEditor.currentFrame.getCurrentMode().activate();
                BlasterEditor.currentFrame.grabFocus();
```

```
                BlasterEditor.currentFrame.moveToFront();
                try
                {
                    BlasterEditor.currentFrame.setSelected(true);
                }
                catch (PropertyVetoException ee)
                {
                }
            }
        });
        polygonModeButton.setSelected(true);
        group.add(polygonModeButton);
        getContentPane().add(polygonModeButton);
        moveModeButton = new JToggleButton("M");
        moveModeButton.addActionListener(new ActionListener()
        {

            public void actionPerformed(ActionEvent e)
            {
                BlasterEditor.currentFrame.getCurrentMode().deactivate();
                BlasterEditor.currentFrame.setCurrentMode(BlasterEditor.currentF
rame.moveMode);
                BlasterEditor.currentFrame.getCurrentMode().activate();
                BlasterEditor.currentFrame.grabFocus();
                BlasterEditor.currentFrame.moveToFront();
                try
                {
                    BlasterEditor.currentFrame.setSelected(true);
                }
                catch (PropertyVetoException ee)
                {
                }
            }
        });
        group.add(moveModeButton);
        getContentPane().add(moveModeButton);

        this.putClientProperty("JInternalFrame.isPalette", Boolean.TRUE);

    }
    public void refresh()
    {
        if (BlasterEditor.currentFrame.getCurrentMode() instanceof PolygonMode)
            polygonModeButton.setSelected(true);
        if (BlasterEditor.currentFrame.getCurrentMode() instanceof MoveMode)
            moveModeButton.setSelected(true);
    }
}
```



```
/**
 * ----------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ----------------------------------------------------------------------------
 */

package dk.ruc.blaster.ui;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Point;
import java.awt.Rectangle;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.MouseEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyVetoException;
import java.beans.VetoableChangeListener;
import java.io.File;

import javax.swing.JInternalFrame;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JViewport;
import javax.swing.event.InternalFrameEvent;
import javax.swing.event.InternalFrameListener;
import javax.swing.event.MouseInputAdapter;

import dk.ruc.blaster.model.EditorObject;
import dk.ruc.blaster.model.Map;

/*
 * dk13043 did not update javadoc, and $Author$ didn't care :)
 *
 * Last change by: $Author$
 *
 * $Header$
 *
 * @version $Revision$
 * @author dk13043
 */
public class MapFrame
    extends JInternalFrame
    implements InternalFrameListener, VetoableChangeListener
{
    Map map;
    private AbstractMode currentMode;
    final public PolygonMode polygonMode;
    final public MoveMode moveMode;
    private String title;
    private JScrollPane scrollView;
    private EditorCanvas canvas;
    private File lastUsedDir = null;

    private EditorObject currentSelection = null;
    /**
     * Constructor for MapFrame.
     */
    public MapFrame(String title)
```

```
    {
        this.title = title;
        map = new Map();
        moveMode = new MoveMode(map);
        polygonMode = new PolygonMode(map);
        init();
    }
    public MapFrame(Map map)
    {
        this.map = map;
        this.title = map.name;
        moveMode = new MoveMode(map);
        polygonMode = new PolygonMode(map);
        init();
    }

    private void init()
    {
        addVetoableChangeListener(this);
        currentMode = polygonMode;
        addInternalFrameListener(this);

        canvas = new EditorCanvas(map);
        canvas.setSize(1024, 768);
        scrollView = new JScrollPane(canvas);
        scrollView.setHorizontalScrollBarPolicy(
            JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        scrollView.setVerticalScrollBarPolicy(
            JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
        scrollView.setMinimumSize(new Dimension(100, 100));
        scrollView.setPreferredSize(new Dimension(320, 240));

        getContentPane().add(scrollView, BorderLayout.CENTER);

        this.setClosable(true);
        this.setMaximizable(true);
        this.setIconifiable(true);
        this.setResizable(true);

        KeyAdapter keyHandler = new KeyAdapter()
        {
            public void keyPressed(KeyEvent e)
            {
                final Rectangle bounds = currentMode.keyPressed(e);

                if (bounds != null)
                    canvas.repaint(
                        bounds.x,
                        bounds.y,
                        bounds.width + 1,
                        bounds.height + 1);
            }

            public void keyReleased(KeyEvent e)
            {
                final Rectangle bounds = currentMode.keyReleased(e);

                if (bounds != null)
                    canvas.repaint(
                        bounds.x,
                        bounds.y,
                        bounds.width + 1,
                        bounds.height + 1);
```



```
            }

            public void keyTyped(KeyEvent e)
            {
                final Rectangle bounds = currentMode.keyTyped(e);

                if (bounds != null)
                    canvas.repaint(
                        bounds.x,
                        bounds.y,
                        bounds.width + 1,
                        bounds.height + 1);
            }
        };

        MouseInputAdapter inputHandler = new MouseInputAdapter()
        {
            public void mousePressed(MouseEvent e)
            {
                final Rectangle bounds = currentMode.mousePressed(e);
                if (bounds != null)
                    canvas.repaint(
                        bounds.x - 1,
                        bounds.y - 1,
                        bounds.width + 2,
                        bounds.height + 2);
            }
            public void mouseReleased(MouseEvent e)
            {
                final Rectangle bounds = currentMode.mouseReleased(e);
                if (bounds != null)
                    canvas.repaint(
                        bounds.x - 1,
                        bounds.y - 1,
                        bounds.width + 2,
                        bounds.height + 2);
            }
            public void mouseClicked(MouseEvent e)
            {
                final Rectangle bounds = currentMode.mouseClicked(e);
                if (bounds != null)
                    canvas.repaint(
                        bounds.x - 1,
                        bounds.y - 1,
                        bounds.width + 2,
                        bounds.height + 2);
            }
            public void mouseDragged(MouseEvent e)
            {
                final Rectangle bounds = currentMode.mouseDragged(e);
                if (bounds != null)
                    canvas.repaint(
                        bounds.x - 1,
                        bounds.y - 1,
                        bounds.width + 2,
                        bounds.height + 2);
            }
            public void mouseMoved(MouseEvent e)
            {
                final Rectangle bounds = currentMode.mouseMoved(e);
                if (bounds != null)
                    canvas.repaint(
                        bounds.x - 1,
```

```
                        bounds.y - 1,
                        bounds.width + 2,
                        bounds.height + 2);
            }
            public void mouseEntered(MouseEvent e)
            {
                final Rectangle bounds = currentMode.mouseEntered(e);
                if (bounds != null)
                    canvas.repaint(
                        bounds.x - 1,
                        bounds.y - 1,
                        bounds.width + 2,
                        bounds.height + 2);
            }
            public void mouseExited(MouseEvent e)
            {
                final Rectangle bounds = currentMode.mouseEntered(e);
                if (bounds != null)
                    canvas.repaint(
                        bounds.x - 1,
                        bounds.y - 1,
                        bounds.width + 2,
                        bounds.height + 2);
            }
        };
        canvas.addMouseListener(inputHandler);
        canvas.addMouseMotionListener(inputHandler);
        canvas.addKeyListener(keyHandler);

    }
    public void display()
    {
        try
        {
            super.setVisible(true);
            this.setMaximum(true);

            final JViewport v = scrollView.getViewport();
            final Dimension canvasSize = canvas.getSize();
            final Rectangle view = v.getViewRect();

            final int vx = (canvasSize.width - view.width) / 2;
            final int vy = (canvasSize.height - view.height) / 2;

            view.setLocation(new Point(vx, vy));

            v.scrollRectToVisible(view);
            canvas.requestFocus();
        }
        catch (Exception e)
        {
        }
    }
    public void switchMode()
    {
        if (currentMode instanceof PolygonMode)
        {
            currentMode.deactivate();
            currentMode = moveMode;
            currentMode.activate();
        }
        else
        {
```



```
            currentMode.deactivate();
            currentMode = polygonMode;
            currentMode.activate();
        }
    }
    /**
     * Returns the currentMode.
     * @return AbstractMode
     */
    public AbstractMode getCurrentMode()
    {
        return currentMode;
    }

    /**
     * Sets the currentMode.
     * @param currentMode The currentMode to set
     */
    public void setCurrentMode(AbstractMode currentMode)
    {
        this.currentMode = currentMode;
    }

    /**
     * @see javax.swing.event.InternalFrameListener#internalFrameActivated(InternalFrameEvent)
     */
    public void internalFrameActivated(InternalFrameEvent e)
    {
        BlasterEditor.currentFrame = this;
        FloatingToolBar.getInstance().refresh();
    }

    /**
     * @see javax.swing.event.InternalFrameListener#internalFrameClosed(InternalFrameEvent)
     */
    public void internalFrameClosed(InternalFrameEvent e)
    {

    }

    /**
     * @see javax.swing.event.InternalFrameListener#internalFrameClosing(InternalFrameEvent)
     */
    public void internalFrameClosing(InternalFrameEvent e)
    {
    }

    /**
     * @see javax.swing.event.InternalFrameListener#internalFrameDeactivated(InternalFrameEvent)
     */
    public void internalFrameDeactivated(InternalFrameEvent e)
    {
    }

    /**
     * @see javax.swing.event.InternalFrameListener#internalFrameDeiconified(InternalFrameEvent)
     */
    public void internalFrameDeiconified(InternalFrameEvent e)
```

```
    {
        BlasterEditor.currentFrame = this;
    }

    /**
     * @see javax.swing.event.InternalFrameListener#internalFrameIconified(InternalFrameEvent)
     */
    public void internalFrameIconified(InternalFrameEvent e)
    {
    }

    /**
     * @see javax.swing.event.InternalFrameListener#internalFrameOpened(InternalFrameEvent)
     */
    public void internalFrameOpened(InternalFrameEvent e)
    {
        BlasterEditor.windowCount++;
        BlasterEditor.currentFrame = this;
    }

    public void vetoableChange(PropertyChangeEvent pce)
        throws PropertyVetoException
    {

        if (pce.getPropertyName().equals(IS_CLOSED_PROPERTY))
        {

            int option =
                JOptionPane.showInternalConfirmDialog(
                    this,
                    "Do you really want to close?",
                    "Warning",
                    JOptionPane.YES_NO_OPTION,
                    JOptionPane.WARNING_MESSAGE);

            if (option != JOptionPane.YES_OPTION)
            {
                throw new PropertyVetoException("User cancelled", pce);
            }

        }
    }
    /**
     * Returns the title.
     * @return String
     */
    public String getTitle()
    {
        return title;
    }

    /**
     * Sets the title.
     * @param title The title to set
     */
    public void setTitle(String title)
    {
        this.title = title;
    }

    /**
```



```
     * Returns the currentSelection.
     * @return EditorObject
     */
    public EditorObject getCurrentSelection()
    {
        return currentSelection;
    }

    /**
     * Sets the currentSelection.
     * @param currentSelection The currentSelection to set
     */
    public void setCurrentSelection(EditorObject currentSelection)
    {
        this.currentSelection = currentSelection;
    }

    /**
     * Returns the lastUsedDir.
     * @return File
     */
    public File getLastUsedDir()
    {
        return lastUsedDir;
    }

    /**
     * Sets the lastUsedDir.
     * @param lastUsedDir The lastUsedDir to set
     */
    public void setLastUsedDir(File lastUsedDir)
    {
        this.lastUsedDir = lastUsedDir;
    }

    /**
     * Returns the canvas.
     * @return EditorCanvas
     */
    public EditorCanvas getCanvas()
    {
        return canvas;
    }
}
```

```
/**
 * ----------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ----------------------------------------------------------------------------
 */

package dk.ruc.blaster.ui;

import java.awt.Point;
import java.awt.Rectangle;
import java.awt.event.MouseEvent;
import java.awt.event.KeyEvent;
import java.util.Iterator;

import dk.ruc.blaster.model.EditorObject;
import dk.ruc.blaster.model.Map;
import dk.ruc.blaster.model.Triangle;
import dk.ruc.blaster.model.Vertex;
import dk.ruc.blaster.ui.property.PropertyFloat;

public class MoveMode extends AbstractMode
{
    private EditorObject selectedObject = null;

    private Point lastPosition;
    private Point startPosition;

    public MoveMode(Map m)
    {
        super(m);
    }

    public void activate()
    {
        if (selectedObject != null)
        {
            selectedObject.select(false);
        }

        System.out.println("MoveMode activated");
    }

    public void deactivate()
    {
        if (selectedObject != null)
        {
            selectedObject.select(false);
        }

        System.out.println("MoveMode deactivated");
    }

    public Rectangle mouseDragged(MouseEvent e)
    {

        final Point currentPosition = e.getPoint();
        final Rectangle r = new Rectangle();
        final int dx = currentPosition.x - lastPosition.x;
        final int dy = currentPosition.y - lastPosition.y;
```



```
        if (selectedObject == null)
        {
            // drag box?
        }
        else
        {
            selectedObject.move(dx, dy);
            Rectangle rr = selectedObject.updateOnDrag();
            if(rr != null)
                r.add(rr);
        }
        r.add(lastPosition);
        r.add(currentPosition);

        lastPosition = currentPosition;
        return r;
    }

    public Rectangle mousePressed(MouseEvent e)
    {

        Point q = e.getPoint();
        EditorObject lastSelected = selectedObject;
        map.unselectall();
        EditorObject v = map.select(q);

        selectedObject = v;

        lastPosition = startPosition = q;

        if (v == null)
        {
            return null;
        }

        v.select(true);
        PropertyFloat.getInstance().setData(v.getProperties());
        BlasterEditor.currentFrame.setCurrentSelection(v);
        /*System.out.println(
            "Mouse pressed, "
                + (selectedObject == null ? "0" : "1")
                + " object(s) selected");*/

        if (lastSelected != null)
        {
            Rectangle r = v.bounds();
            r.add(lastSelected.bounds());
            return r;
        }
        else
            return v.bounds();

    }

    public Rectangle mouseReleased( MouseEvent e )
    {
        if (selectedObject instanceof Vertex)
        {
            Vertex v = (Vertex) selectedObject;
            final Point currentPosition = e.getPoint();
            Vertex nearestVertex = map.getNearestVertexToPoint(currentPosition,
v);
```

```
            if (nearestVertex == null)
            {
                return null;
            }

            if ( v.bounds().intersects( nearestVertex.bounds() ) )
            {
                boolean update = false;

                Triangle t;
                Iterator i = map.triangles.iterator();

                while (i.hasNext())
                {
                    t = (Triangle) i.next();

                    for( int i2 = 0; i2 < t.edges.length; i2++)
                    {
                        if (v == t.edges[i2].getA())
                        {
                            t.edges[i2].setA(nearestVertex);
                            update = true;
                        }
                    }

                    for( int i2 = 0; i2 < t.vertices.length; i2++)
                    {
                        if (v == t.vertices[i2])
                        {
                            t.vertices[i2] = nearestVertex;
                            update = true;
                        }
                    }

                    if ( update )
                    {
                        nearestVertex.addPoly(t);
                        nearestVertex.updateOnDrag();
                    }
                }
            map.vertices.remove(v);
            }
        }
    return null;
    }

    /**
     * @see dk.ruc.blaster.ui.AbstractMode#keyPressed(KeyEvent)
     */
    public Rectangle keyPressed(KeyEvent e)
    {
    if (e.getKeyCode() == KeyEvent.VK_DELETE)
        {
            if (selectedObject instanceof Triangle)
            {
                Triangle t = (Triangle) selectedObject;
                    for (int i2 = 0; i2 < t.vertices.length; i2++)
                    {
                        t.vertices[i2].removePoly( t );

                        if (t.vertices[i2].getNumberOfAttachedPolygons() == 0)
                        {
                            map.vertices.remove(t.vertices[i2]);
```



```
                        }
                    }
                }
                map.triangles.remove(selectedObject);

            }
            return selectedObject.bounds();
        }

        /**
         * @see dk.ruc.blaster.ui.AbstractMode#keyReleased(KeyEvent)
         */
        public Rectangle keyReleased(KeyEvent e) {
            System.out.println("released: "+e.getKeyCode());
            return super.keyReleased(e);
        }

}
```

```
/**
 * ------------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ------------------------------------------------------------------------------
 */

package dk.ruc.blaster.ui;

import java.awt.Rectangle;
import java.awt.event.MouseEvent;

import dk.ruc.blaster.model.Map;
import dk.ruc.blaster.model.Triangle;
import dk.ruc.blaster.model.Vertex;

public class PolygonMode extends AbstractMode
{
    private Vertex verts[];
    private int vertIndex;

    public PolygonMode( Map m )
    {
        super( m );
        reset();
    }

    private void reset()
    {
        verts = new Vertex[]{ null, null, null };
        vertIndex = 0;
    }

    public void activate()
    {
        reset();
    }

    public void deactivate()
    {
        reset();
    }

    public Rectangle mouseClicked( MouseEvent e )
    {
        System.err.println("clicked "+vertIndex);
        // Place a vertex where the user clicked. If the user clicked on an pree
xisting vertex,
        // it will be returned.
        final Vertex v = map.addVertex( e.getPoint() );

        // The part of the window that needs updating when we're done
        final Rectangle bounds = v.bounds();

        // Add the current vertex to our list of vertices
        verts[vertIndex++] = v;

        // When we reach three vertices, create a polygon and reset the list
        if ( vertIndex == 3 )
        {
```



```
            final Triangle t = map.addTriangle( verts );

            bounds.add( t.bounds() );

            reset();
        }

        return bounds;
    }
}
```

```
/**
 * ----------------------------------------------------------------------------
 * "THE BEER-WARE LICENSE" (Revision 42):
 * <doktor@dyregod.dk> <mads@danquah.dk> <tkrogh@ruc.dk> <tnjr@ruc.dk> wrote
 * this file. As long as you retain this notice you can do whatever you want
 * with this stuff. If we meet some day, and you think this stuff is worth it,
 * you can buy us a beer in return.
 * ----------------------------------------------------------------------------
 */

package dk.ruc.blaster.ui.property;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;

import javax.swing.DefaultCellEditor;
import javax.swing.JInternalFrame;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.AbstractTableModel;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableCellEditor;
import javax.swing.table.TableColumn;

import dk.ruc.blaster.ui.BlasterEditor;

/*
 * dk13043 did not update javadoc, and $Author$ didn't care :)
 *
 * Last change by: $Author$
 *
 * $Header$
 *
 * @version $Revision$
 * @author dk13043
 */
public class PropertyFloat extends JInternalFrame
{
    private static PropertyFloat instance = null;
    JTable table;
    KeyValueTableData data;
    /**
     * Constructor for PropertyFloat.
     */
    private PropertyFloat()
    {
        super();
        this.putClientProperty("JInternalFrame.isPalette", Boolean.TRUE);
        setSize(120, 130);
        setLocation(20, 160);
        data = new KeyValueTableData(null);
        table = new JTable();
        table.setAutoCreateColumnsFromModel(false);
        data.setDefaultData();
        table.setModel(data);

        for (int k = 0; k < KeyValueTableData.m_columns.length; k++)
        {
            DefaultTableCellRenderer renderer = new DefaultTableCellRenderer();
```



```
            renderer.setHorizontalAlignment(
                KeyValueTableData.m_columns[k].m_alignment);

            TableCellEditor editor = null;

            if (k == 1)
            {
                editor = new DefaultCellEditor(new JTextField());
            }

            TableColumn column =
                new TableColumn(
                    k,
                    KeyValueTableData.m_columns[k].m_width,
                    renderer,
                    editor);
            table.addColumn(column);

        }

        JTableHeader header = table.getTableHeader();
        header.setUpdateTableInRealTime(true);

        JScrollPane ps = new JScrollPane();
        ps.getViewport().add(table);

        this.getContentPane().add(ps);

        this.setVisible(true);

    }
    public void setData(HashMap map)
    {
        System.err.println("huzaaa" + map);
        data = new KeyValueTableData(map);
        data.setDefaultData();
        table.setModel(data);

    }
    public static PropertyFloat getInstance()
    {
        if (instance == null)
        {
            instance = new PropertyFloat();
        }
        return instance;
    }
    public void refresh()
    {
    }
}
class KeyValueData
{
    public String key;
    public String value;

    public KeyValueData(String key, String value)
    {
        this.key = key;
        this.value = value;

    }
}
```

```
class ColumnData
{
    public String m_title;
    public int m_width;
    public int m_alignment;

    public ColumnData(String title, int width, int alignment)
    {
        m_title = title;
        m_width = width;
        m_alignment = alignment;
    }
}
class KeyValueTableData extends AbstractTableModel
{
    static final public ColumnData m_columns[] =
        {
            new ColumnData("Key", 25, JLabel.LEFT),
            new ColumnData("Value", 40, JLabel.LEFT)};

    ArrayList list = new ArrayList();

    HashMap map;

    public KeyValueTableData(HashMap map)
    {
        this.map = map;
    }

    public void setDefaultData()
    {
        list.clear();
        if (map == null)
            return;
        for (Iterator iterator = map.keySet().iterator(); iterator.hasNext();)
        {
            String element = (String) iterator.next();
            String value = (String) map.get(element);
            list.add(new KeyValueData(element, value));
        }

    }

    public int getRowCount()
    {
        return list == null ? 0 : list.size();
    }

    public int getColumnCount()
    {
        return m_columns.length;
    }

    public String getColumnName(int column)
    {
        return m_columns[column].m_title;
    }

    public boolean isCellEditable(int nRow, int nCol)
    {
        if (nCol == 1)
            return true;
```



```
        else
            return false;
    }

    public Object getValueAt(int nRow, int nCol)
    {
        if (nRow < 0 || nRow >= getRowCount())
            return null;
        KeyValueData row = (KeyValueData) list.get(nRow);
        if (nCol == 0)
            return row.key;
        else
            return row.value;
    }

    public String getTitle()
    {
        return "Properties";
    }
    /**
     * @see javax.swing.table.TableModel#setValueAt(Object, int, int)
     */
    public void setValueAt(Object aValue, int rowIndex, int columnIndex)
    {
        list.set(rowIndex, new KeyValueData( ((KeyValueData)list.get(rowIndex)).
key, (String) aValue));
        HashMap hash = new HashMap();

        for (Iterator iter = list.iterator(); iter.hasNext();)
        {
            KeyValueData d = (KeyValueData) iter.next();
            hash.put(d.key, d.value);
        }

        if (BlasterEditor.currentFrame.getCurrentSelection()== null)
            return;
        BlasterEditor.currentFrame.getCurrentSelection().setProperties(hash);
    }

}
```