WorldWind

1 Code

```
/*******************************************************
*
*
*       WORLDWINDMAP for Marker und Flyanimation
*
*       Author:             René Krause
*       Version:            2D, Sphere and 3D
*       Date:               15.06.2016
*       Class 'map':        52 Functions
*       Lines Of Code:      918
*
*       Listener:           - Action
*                           - Mouse
*                           - MouseWheel
*                           - Components
*                           - Key
*                           - GLEvent
*
*       Status:             - Development      OK
*                           - Testing          OK
*                           - final Version    OK
*
*
*
*
*
*******************************************************/
package gov.nasa.worldwind;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import javax.media.opengl.*;
import javax.swing.Timer;
import javax.imageio.ImageIO;
import java.util.LinkedList;
import java.util.ArrayList;
import gov.nasa.worldwind.*;
import gov.nasa.worldwind.globes.*;
import gov.nasa.worldwind.awt.*;
import gov.nasa.worldwind.avlist.*;
import gov.nasa.worldwind.event.*;
import gov.nasa.worldwind.render.*;
import gov.nasa.worldwind.util.StatusBar;
import gov.nasa.worldwind.layers.RenderableLayer;
import gov.nasa.worldwind.layers.SurfaceImageLayer;
import gov.nasa.worldwind.render.Polyline;
import gov.nasa.worldwind.render.SurfaceImage;
import gov.nasa.worldwind.render.ScreenImage;
import gov.nasa.worldwind.render.Ellipsoid;
import gov.nasa.worldwind.render.PointPlacemark;
import gov.nasa.worldwind.geom.Angle;
import gov.nasa.worldwind.geom.Sector;
import gov.nasa.worldwind.geom.LatLon;
import gov.nasa.worldwind.geom.Position;
import gov.nasa.worldwind.geom.Vec4;
import gov.nasa.worldwind.geom.Sphere;
public class worldwind{
public static void main(String[] args){
SwingUtilities.invokeLater(
new Runnable(){
               @Override
               public void run(){
           JFrame frame=new map();
           Container contenPane=frame.getContentPane();
           frame.setSize(1000,800);
          frame.setVisible(true);
               }
    }
);
}
}
class map extends JFrame implements ActionListener{
```

```
public WorldWindowGLCanvas        wwd;
public ActionListener             timer;
public JPanel                     panel;
public JLabel                     label,l1,l2,l3,l4,l5,l6,l7,l8,l9,l10,l11,l12,l13,l14,l15,l16,l17,l18,l19;
public JScrollPane                scrollPane;
public JButton                    button;
public JTextField                 t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,t15;
public ButtonGroup                group;
public JRadioButton               r1,r2,r3,r4,r5;
public ScreenImage                navi1,navi2;
private LinkedList<Position>      markerList,areaList;
    private RenderableLayer           layerS,layerPL;
/*****************************************************
*
*  Path: set your own path
*
*
*****************************************************/
private String Path=              "C:/Users/User/Desktop/eclipse/workspace/worldwind6/";
/*****************************************************
*
*****************************************************/
private double[][]                markerArray;
private double[][]                areaArray;
private int Index=                -1;
    private int modus=                2;
    private int timerSpeed=           100;
    private int counter=              0;
private boolean navi=             true;
private boolean settings=         false;
private boolean plane=            false;
private boolean firstPerson=      false;
private boolean externFly=        false;
private double homeLatitude=      51.14906345685532;
private double homeLongitude=     14.997722278665057;
private double homeElevation=     5000.0;
private double homePitch=         45.0;
private double homeHeading=       0.0;
private Color  polylineColor=     Color.RED;
private double polylineLineWidth= 3.0;
private double planeLatitude=     51.14906345685532;
private double planeLongitude=    14.997722278665057;
private double planeElevation=    500.0;
private double planeRadius=       30.0;
private double markerElevation=   100.0;
    private int    rotor=             0;
/*****************************************************
*
*  map with init functions
*
*
*****************************************************/
public map(){
        markerList= new LinkedList<Position>();
        areaList=   new LinkedList<Position>();
        initDoubleArrays();
// init functions
    initTimer();
    new Timer(timerSpeed,timer).start();
initGLCanvas(new WorldWindowGLCanvas());
initJPanel();
initStatusBar(new StatusBar());
    initScreenImage(Path+"img/");//navi
    initMouseListener();
    initMouseWheelListener();
    initKeyListener();
    initGLEventListener();
    initComponentListener();
}
/*****************************************************
*
*  navi with events = 10 functions
*
*
*****************************************************/
private boolean navi(int x,int y){
```

```
boolean check=false;
if((x>10)&&(y<70)){
     int begin,modi;
     if(navi==true){begin=280;modi=48;}else{begin=374;modi=73;}
     int end=begin+50;
         for(int i=0;i<9;i++){
          if((x>(begin+(modi*i)))&&((x<(end+(modi*i))))){
          check=true;
          firstPerson=false;
          plane=false;
          Index=-1;
          if(settings==true){settings=setStatusElementsVisible(false);}
            panel.setVisible(false);
          scrollPane.setVisible(false);
          l10.setVisible(false);
          switch(i){
          case 0:homeState();break;
          case 1:globeState();break;
          case 2:markerState();break;
             case 3:areaState();break;
         case 4:flyState();break;
         case 5:cameraState();break;
             case 6:statusState();break;
           case 7:settingsState();break;
             case 8:creditState();break;
          }
          }
          }
      }
return check;
}
private void homeState(){
System.out.println("homepoint");
setGoToLocation(homeLatitude,homeLongitude,homeElevation,homePitch,homeHeading);
}
private void globeState(){
System.out.println("globe");
setGoToLocation(30.0,10.0,20000000.0,0.0,0.0);
}
    private void markerState(){
    System.out.println("marker");
    Index=0;
}
private void areaState(){
    System.out.println("area");
    Index=1;
}
private void flyState(){
System.out.println("fly");
plane=true;
double pitch=45.0;
if(modus==0){pitch=0.0;}
setGoToLocation(planeLatitude,planeLongitude,4000.0,pitch,0.0);
}
private void cameraState(){
System.out.println("camera");
firstPerson=true;
counter=40;
setGoToLocation(planeLatitude,planeLongitude,4000.0,80.0,90.0);
}
    private void statusState(){
System.out.println("status");
panel.setVisible(true);
scrollPane.setVisible(true);
label.setText(getStatusText());
}
private void settingsState(){
System.out.println("settings");
settings=true;
   panel.setVisible(true);
boolean bool=setStatusElementsVisible(true);
}
private void creditState(){
System.out.println("credit");
   panel.setVisible(true);
   l10.setVisible(true);
```

```
   l10.setText(""+getCreditText());
}
/*******************************************************
*
*  public = 15 functions
*
*
*******************************************************/
public double[][] getMarker(){return this.markerArray;}
public double[][] getArea()  {return this.areaArray;}
    public LinkedList<Position> getMarkerList(){return this.markerList;}
    public LinkedList<Position> getAreaList()  {return this.areaList;}
    public double[] getHomePoint(){
     double[] homePoint=new double[5];
     homePoint[0]=this.homeLatitude;
     homePoint[1]=this.homeLongitude;
     homePoint[2]=this.homeElevation;
     homePoint[3]=this.homePitch;
     homePoint[4]=this.homeHeading;
     return homePoint;
    }
    public double[] getPlane(){
     double[] plane=new double[3];
     plane[0]=planeLatitude;
     plane[1]=planeLongitude;
     plane[2]=planeElevation;
     return plane;
    }
    public void setPosition(double lat,double lon,double ele,double pitch,double head){
    Position pos=Position.fromDegrees(lat,lon,ele);
    wwd.getView().setEyePosition(pos);
    wwd.getView().setPitch(Angle.fromDegrees(pitch));
    wwd.getView().setHeading(Angle.fromDegrees(head));
    }
    public void setGoToLocation(double lat,double lon,double ele,double pitch,double head){
    wwd.getView().setPitch(Angle.fromDegrees(pitch));
    wwd.getView().setHeading(Angle.fromDegrees(head));
     wwd.getView().goTo(Position.fromDegrees(lat,lon,ele),ele);
    }
    public void setHomePoint(double lat,double lon,double ele,double pitch,double head){
homeLatitude=lat;
homeLongitude=lon;
homeElevation=ele;
homePitch=pitch;
homeHeading=head;
    }
    public void setMarker(double lat,double lon,double ele){
setMarkerAreaArray(lat,lon,ele,0,true);
        markerList.add(Position.fromDegrees(lat,lon,ele));
        String filepath=Path+"img/marker.gif";
        lat=lat-0.00000030;
        lon=lon-0.000501;
        Angle minLon=Angle.fromDegrees(lon);
        Angle maxLon=Angle.fromDegrees(lon+0.001);
        Angle minLat=Angle.fromDegrees(lat);
        Angle maxLat=Angle.fromDegrees(lat+0.0015);
        Sector sector=new Sector(minLat,maxLat,minLon,maxLon);
        SurfaceImage surfaceImage=new SurfaceImage(filepath,sector);
        surfaceImage.setOpacity(1.0);
        SurfaceImageLayer layerI=new SurfaceImageLayer();
        layerI.setPickEnabled(false);
        layerI.addRenderable(surfaceImage);
    wwd.getModel().getLayers().add(layerI);
    wwd.redraw();
}
    public void setSphere(Vec4 vec,double lat,double lon,double ele){
     ele=ele+markerElevation+200.0;
setMarkerAreaArray(lat,lon,ele,0,true);
        markerList.add(Position.fromDegrees(lat,lon,ele));
        Sphere sphere=new Sphere(vec,planeRadius);
        RenderableLayer layerPS=new RenderableLayer();
        layerPS.setPickEnabled(false);
        layerPS.addRenderable(sphere);
       wwd.getModel().getLayers().add(layerPS);
       wwd.redraw();
    }
```

```
    public void setEllipsoid(double lat,double lon,double ele){
     ele=ele+markerElevation;
setMarkerAreaArray(lat,lon,ele,0,true);
        markerList.add(Position.fromDegrees(lat,lon,ele));
     Ellipsoid el=new Ellipsoid(Position.fromDegrees(lat,lon,ele),15.0,30.0,10.0);
     el.setImageSources(Path+"img/ellipsoid_marker.gif");
     RenderableLayer layerPE=new RenderableLayer();
        layerPE.setPickEnabled(true);
        layerPE.addRenderable(el);
       wwd.getModel().getLayers().add(layerPE);
       wwd.redraw();
    }
    public void setArea(double lat,double lon,double ele){
     if(modus!=0){ele=ele+markerElevation;}
setMarkerAreaArray(lat,lon,ele,0,true);
     areaList.add(Position.fromDegrees(lat,lon,ele));
Polyline polyline=new Polyline(areaList);
polyline.setColor(polylineColor);
polyline.setLineWidth(polylineLineWidth);
RenderableLayer layerP=new RenderableLayer();
layerP.addRenderable(polyline);
wwd.getModel().getLayers().add(layerP);
wwd.redraw();
    }
    public void setPlaneValues(double lat,double lon,double ele,double rad){
     externFly=true;
     planeLatitude=lat;
     planeLongitude=lon;
     planeElevation=ele;
     planeRadius=rad;
    }
    public void setPlaneValuesFromSettings(double ele,double rad){
planeElevation=ele;
planeRadius=rad;
    }
/*******************************************************
*
*  private = 8 functions
*
*
*******************************************************/
    private boolean setStatusElementsVisible(boolean visible){
button.setVisible(visible);
t1.setVisible(visible);
t2.setVisible(visible);
t3.setVisible(visible);
t4.setVisible(visible);
t5.setVisible(visible);
t6.setVisible(visible);
t7.setVisible(visible);
t8.setVisible(visible);
t9.setVisible(visible);
t10.setVisible(visible);
t11.setVisible(visible);
r1.setVisible(visible);
r2.setVisible(visible);
r3.setVisible(visible);
l1.setVisible(visible);
l2.setVisible(visible);
l3.setVisible(visible);
  l4.setVisible(visible);
l5.setVisible(visible);
l6.setVisible(visible);
  l7.setVisible(visible);
l8.setVisible(visible);
l9.setVisible(visible);
l10.setVisible(visible);
l11.setVisible(visible);
l12.setVisible(visible);
l13.setVisible(visible);
l14.setVisible(visible);
l15.setVisible(visible);
l16.setVisible(visible);
l17.setVisible(visible);
return false;
    }
```

```
private String getStatusText(){
String str="<html>Marker<br>";
for(int i=0;i<99;i++){
if(markerArray[i][0]!=0.0){
    str=str+""+i+" lat: "+markerArray[i][0]+"<br />";
    str=str+">> lon: "+markerArray[i][1]+"<br />";
    str=str+">> ele: "+markerArray[i][2]+"<br /><br />";
}
}
str=str+"Area<br />";
for(int i=0;i<99;i++){
if(areaArray[i][0]!=0.0){
    str=str+""+i+" lat: "+areaArray[i][0]+"<br />";
    str=str+">> lon: "+areaArray[i][1]+"<br />";
    str=str+">> ele: "+areaArray[i][2]+"<br /><br />";
}
}
return str+"</html>";
}
private String getCreditText(){
String str="<html>Credits<br>";
str=str+"http://worldwind.arc.nasa.gov/<br />java/index.html<br />";
str=str+"https://goworldwind.org<br />";
str=str+"http://forum.<br >worldwindcentral.com";
return str+"</html>";
}
private void setNaviOpacity(){
int boundWidth=wwd.getBounds().width;
int imageX0=navi1.getScreenLocation().x;
int imageX1=navi2.getScreenLocation().x;
if((boundWidth>1000)&&(imageX0<500)){
navi1.setOpacity(0.0);
navi2.setOpacity(1.0);
navi=false;
}
  if((boundWidth<1000)&&(imageX1>500)){
navi1.setOpacity(1.0);
navi2.setOpacity(0.0);
navi=true;
}
}
private void setMarkerAreaArray(double lat,double lon,double ele,int mod,boolean check){
for(int i=0;i<99;i++){
if(check==true){
if(mod==0){
        if(markerArray[i][0]==0.0){
         markerArray[i][0]=lat;
         markerArray[i][1]=lon;
         markerArray[i][2]=ele;
         check=false;
        }
}
    if(mod==1){
        if(areaArray[i][0]==0.0){
         areaArray[i][0]=lat;
         areaArray[i][1]=lon;
         areaArray[i][2]=ele;
         check=false;
        }
    }
}
}
}
    private void setPolylineValues(String colorStr,double lineWidth){
     Color color;
     switch(colorStr){
          case "Color.WHITE":polylineColor=Color.WHITE;break;
          case "Color.BLACK":polylineColor=Color.BLACK;break;
          case "Color.RED":polylineColor=Color.RED;break;
          case "Color.BLUE":polylineColor=Color.BLUE;break;
          case "Color.GREEN":polylineColor=Color.GREEN;break;
          case "Color.YELLOW":polylineColor=Color.YELLOW;break;
          case "Color.GRAY":polylineColor=Color.GRAY;break;
          case "Color.ORANGE":polylineColor=Color.ORANGE;break;
          case "Color.MAGENTA":polylineColor=Color.MAGENTA;break;
          case "Color.PINK":polylineColor=Color.PINK;break;
```

```
     }
     polylineLineWidth=lineWidth;
    }
    private void setPlane(){
     if(layerPL!=null){wwd.getModel().getLayers().remove(layerPL);}
     if(externFly==false){planeLongitude=planeLongitude-0.00003;}
     double lat=planeLatitude;
     double lon=planeLongitude;
     double ele=planeElevation;
     switch(modus){
         case 0:
         if(rotor==7){rotor=0;}else{rotor++;}
             String filepath = Path+"img/drohn"+rotor+".gif";
             planeLongitude=planeLongitude-0.00003;
             Angle minLon=Angle.fromDegrees(lon);
             Angle maxLon=Angle.fromDegrees(lon+0.004);
             Angle minLat=Angle.fromDegrees(lat);
             Angle maxLat=Angle.fromDegrees(lat+0.004);
             Sector sector=new Sector(minLat,maxLat,minLon,maxLon);
             SurfaceImage plane=new SurfaceImage(filepath,sector);
             layerPL = new SurfaceImageLayer();
             layerPL.setPickEnabled(false);
                layerPL.addRenderable(plane);
     break;
         case 1:
                Globe globe=wwd.getView().getGlobe();
             LatLon pos=new LatLon(Angle.fromDegrees(lat),Angle.fromDegrees(lon));
                Vec4 vec=globe.computePointFromPosition(pos,ele);
                Sphere sphere=new Sphere(vec,50);
                layerPL=new RenderableLayer();
                layerPL.setPickEnabled(true);
                layerPL.addRenderable(sphere);
         break;
         case 2:
           Ellipsoid el=new Ellipsoid(Position.fromDegrees(lat,lon,ele),30.0,5.0,30.0);
             el.setImageSources(Path+"img/ellipsoid_drohn.gif");
             layerPL=new RenderableLayer();
                layerPL.setPickEnabled(true);
                layerPL.addRenderable(el);
          break;
     }
       wwd.getModel().getLayers().add(layerPL);
       wwd.redraw();
    }
    private void firstPersonSetPosition(){
     planeLatitude=planeLatitude-0.00005;
     setPosition(planeLatitude,planeLongitude,planeElevation+90.0,0.0,90.0);
    }
/*******************************************************
*
*  timer
*
*
*******************************************************/
    public void initTimer(){
timer=new ActionListener(){
public void actionPerformed(ActionEvent evt){
    if(firstPerson==false){
     if(plane==true){setPlane();}
    }
    else{
     setPlane();
     if(counter>0){counter--;}
     else{firstPersonSetPosition();}
     if(counter==1){System.out.println("firstpersonfly starts");}
    }
}
};
}
/*******************************************************
*
*  actionlistener
*
*
*******************************************************/
    public void actionPerformed(ActionEvent evt){
```

```
     if(evt.getSource()==this.button){
     double lat=Double.parseDouble(t1.getText());
     double lon=Double.parseDouble(t2.getText());
     double ele=Double.parseDouble(t3.getText());
     double pitch=Double.parseDouble(t4.getText());
     double head=Double.parseDouble(t5.getText());
     double lineWidth=Double.parseDouble(t7.getText());
     double planeEle=Double.parseDouble(t8.getText());
     double planeRad=Double.parseDouble(t9.getText());
     Path=t10.getText();
     double markerHeight=Double.parseDouble(t11.getText());
     setHomePoint(lat,lon,ele,pitch,head);
     setPolylineValues(t6.getText(),lineWidth);
     setPlaneValuesFromSettings(planeEle,planeRad);
     if(r1.isSelected()){modus=0;}
     if(r2.isSelected()){modus=1;}
     if(r3.isSelected()){modus=2;}
     markerElevation=markerHeight;
     }
    }
/*****************************************************
*
*  init listener = 5 listener
*
*
*****************************************************/
public void initMouseListener(){
    wwd.addMouseListener(new MouseAdapter(){
         public void mouseClicked(MouseEvent evt){
          if(evt.getSource() instanceof WorldWindowGLCanvas){
           WorldWindowGLCanvas gl=(WorldWindowGLCanvas) evt.getSource();
           int xPos=gl.getMousePosition().x;
           int yPos=gl.getMousePosition().y;
           firstPerson=false;
           boolean check=navi(xPos,yPos);
           if(check==false){
                     double lat=gl.getCurrentPosition().getLatitude().degrees;
                 double lon=gl.getCurrentPosition().getLongitude().degrees;
                 double ele=gl.getCurrentPosition().elevation;
               switch(Index){
                  case 0:
                   switch(modus){
                     case 0:setMarker(lat,lon,ele);break;
                     case 1:
                                     Globe globe=wwd.getView().getGlobe();
                                     LatLon pos=new LatLon(Angle.fromDegrees(lat),Angle.fromDegrees(lon));
                                     Vec4 vec=globe.computePointFromPosition(pos,300.0);
                                     setSphere(vec,lat,lon,ele);
                     break;
                     case 2:setEllipsoid(lat,lon,ele);break;
                   }
                  break;
                  case 1:setArea(lat,lon,ele);break;
               }
           }
             }
         }
         public void mousePressed(MouseEvent evt){}
         public void mouseReleased(MouseEvent evt){}
         public void mouseEntered(MouseEvent evt){}
         public void mouseExited(MouseEvent evt){}
         public void mouseWheelMoved(MouseWheelEvent evt){}
         public void mouseDragged(MouseEvent evt){}
         public void mouseMoved(MouseEvent evt){}
    });
}
public void initMouseWheelListener(){
    wwd.addMouseWheelListener(new MouseWheelListener(){
     public void mouseWheelMoved(MouseWheelEvent evt){
          if(evt.getSource() instanceof WorldWindowGLCanvas){firstPerson=false;}
     }
    });
}
public void initKeyListener(){
    wwd.addKeyListener(new KeyEventState(){
        public void keyPressed(KeyEvent evt){
```

```
            if(evt.getKeyCode()==KeyEvent.VK_A){areaState();}
            if(evt.getKeyCode()==KeyEvent.VK_C){creditState();}
            if(evt.getKeyCode()==KeyEvent.VK_D){wwd.shutdown();}
            if(evt.getKeyCode()==KeyEvent.VK_F){flyState();}
            if(evt.getKeyCode()==KeyEvent.VK_G){globeState();}
            if(evt.getKeyCode()==KeyEvent.VK_H){homeState();}
            if(evt.getKeyCode()==KeyEvent.VK_M){markerState();}
                if(evt.getKeyCode()==KeyEvent.VK_P){
             PointPlacemark placemark=new PointPlacemark(Position.fromDegrees(homeLatitude,homeLongitude,homeElevation));
             placemark.setLabelText("placemark");
             placemark.setLineEnabled(false);
             PointPlacemarkAttributes attrs=new PointPlacemarkAttributes();
             attrs.setImageAddress(Path+"img/marker.gif");
             attrs.setImageOffset(new Offset(0.5,0.5,AVKey.FRACTION,AVKey.FRACTION));
                attrs.setScale(0.1);
             placemark.setAttributes(attrs);
                RenderableLayer layerPS=new RenderableLayer();
                layerPS.setPickEnabled(false);
                layerPS.addRenderable(placemark);
               wwd.getModel().getLayers().add(layerPS);
               wwd.redraw();
            }
            if(evt.getKeyCode()==KeyEvent.VK_S){settingsState();}
            if(evt.getKeyCode()==KeyEvent.VK_R){cameraState();}
            if(evt.getKeyCode()==KeyEvent.VK_T){statusState();}
        }
    });
}
public void initGLEventListener(){
wwd.addGLEventListener(new GLEventListener(){
     public void init(GLAutoDrawable arg0){System.out.println("init");}
     public void reshape(GLAutoDrawable arg0, int arg1, int arg2, int arg3, int arg4){}
        public void dispose(GLAutoDrawable arg0){}
     public void display(GLAutoDrawable arg0){}
    });
}
public void initComponentListener(){
    wwd.addComponentListener(new ComponentListener(){
    public void componentHidden(ComponentEvent arg0){}
    public void componentMoved(ComponentEvent arg0){}
    public void componentResized(ComponentEvent arg0){setNaviOpacity();}
    public void componentShown(ComponentEvent arg0){}
    });
}
/*******************************************************
*
*  init mapcomponents = 10 functions
*
*
*******************************************************/
private void initGLCanvas(WorldWindowGLCanvas gl){
wwd=gl;
    wwd.setPreferredSize(new Dimension(1000,800));
    this.getContentPane().add(wwd,BorderLayout.CENTER);
    BasicModel model=new BasicModel();
    wwd.setModel(model);
}
    private void initDoubleArrays(){
     markerArray=new double[99][3];
        areaArray=new double[99][3];
     for(int i=0;i<99;i++){markerArray[i][0]=0.0;areaArray[i][0]=0.0;}
    }
private void initStatusBar(StatusBar statusBar){
    this.getContentPane().add(statusBar,BorderLayout.PAGE_END);
    statusBar.setEventSource(wwd);
}
private void initScreenImage(String pathToImage){
navi1=new ScreenImage();
    try{navi1.setImageSource(ImageIO.read(new File(pathToImage+"navi0.gif")));}
    catch(IOException ex){}
    navi1.setScreenLocation(new Point(500,45));
    navi1.setOpacity(1.0);
    RenderableLayer layerS0=new RenderableLayer();
        layerS0.setName("Navi");
        layerS0.addRenderable(navi1);
        wwd.getModel().getLayers().add(layerS0);
```

```
navi2=new ScreenImage();
    try{navi2.setImageSource(ImageIO.read(new File(pathToImage+"navi1.gif")));}
    catch(IOException ex){}
    navi2.setScreenLocation(new Point(720,45));
    navi2.setOpacity(0.0);
        RenderableLayer layerS1=new RenderableLayer();
        layerS1.setName("Navi");
        layerS1.addRenderable(navi2);
        wwd.getModel().getLayers().add(layerS1);
}
private void initJPanel(){
    panel=new JPanel(new FlowLayout());
    panel.setPreferredSize(new Dimension(200,730));
    panel.setBackground(new Color(30,30,30));
    panel.setVisible(false);
    group=new ButtonGroup();
    Font font=new Font("Arial",Font.BOLD,13);
    Font font0=new Font("System",Font.BOLD,11);
    Font font1=new Font("System",Font.PLAIN,9);
    Color white=Color.WHITE;
    label=new JLabel("");
    label.setForeground(Color.BLACK);
    label.setFont(font1);
    scrollPane=new JScrollPane(label,ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,ScrollPaneConstants.HORIZONTAL_SCROLLBAR_A
    scrollPane.setPreferredSize(new Dimension(200,730));
    scrollPane.setVisible(false);
    scrollPane.setAlignmentY(Component.TOP_ALIGNMENT);
    panel.add(scrollPane);
    button=new JButton("UPDATE");
    button.setPreferredSize(new Dimension(150,20));
    button.setVisible(false);
    button.setFont(font);
    button.addActionListener(this);
    panel.add(button);
    setGeneralPanel(font,font0,font1,white);
    setHomePointPanel(font,font0,font1,white);
    setAreaPanel(font,font0,font1,white);
    setPlanePanel(font,font0,font1,white);
    setCreditPanel(font0,white);
    this.getContentPane().add(panel,BorderLayout.WEST);
}
private void setGeneralPanel(Font font0,Font font1,Font font2,Color color){
l15=new JLabel("GENERAL");
    l15.setPreferredSize(new Dimension(150,18));
    l15.setForeground(color);
    l15.setVisible(false);
    l15.setFont(font0);
    panel.add(l15);
    l14=new JLabel("MODUS");
    l14.setPreferredSize(new Dimension(150,18));
    l14.setForeground(color);
    l14.setVisible(false);
    l14.setFont(font1);
    panel.add(l14);
    r1=new JRadioButton("2D");
    r1.setPreferredSize(new Dimension(150,18));
    r1.setBackground(new Color(30,30,30));
    r1.setForeground(color);
    r1.setVisible(false);
    r1.setFont(font1);
    r1.addActionListener(this);
    group.add(r1);
    panel.add(r1);
    r2=new JRadioButton("Sphere");
    r2.setPreferredSize(new Dimension(150,18));
    r2.setBackground(new Color(30,30,30));
    r2.setForeground(color);
    r2.setVisible(false);
    r2.setFont(font1);
    r2.addActionListener(this);
    group.add(r2);
    panel.add(r2);
    r3=new JRadioButton("3D",true);
    r3.setPreferredSize(new Dimension(150,18));
    r3.setBackground(new Color(30,30,30));
    r3.setForeground(color);
```

```
    r3.setVisible(false);
    r3.setFont(font1);
    r3.addActionListener(this);
    group.add(r3);
    panel.add(r3);
    l16=new JLabel("PATH");
    l16.setPreferredSize(new Dimension(150,18));
    l16.setForeground(color);
    l16.setVisible(false);
    l16.setFont(font1);
    panel.add(l16);
    t10=new JTextField(""+Path);
    t10.setPreferredSize(new Dimension(150,18));
    t10.setVisible(false);
    t10.setFont(font2);
    panel.add(t10);
}
private void setHomePointPanel(Font font0,Font font1,Font font2,Color color){
    l1=new JLabel("HOMEPOINT");
    l1.setPreferredSize(new Dimension(150,18));
    l1.setForeground(color);
    l1.setVisible(false);
    l1.setFont(font0);
    panel.add(l1);
    l2=new JLabel("LATITUDE");
    l2.setPreferredSize(new Dimension(150,18));
    l2.setForeground(color);
    l2.setVisible(false);
    l2.setFont(font1);
    panel.add(l2);
    t1=new JTextField(""+homeLatitude);
    t1.setPreferredSize(new Dimension(150,18));
    t1.setVisible(false);
    t1.setFont(font2);
    panel.add(t1);
    l3=new JLabel("LONGITUDE");
    l3.setPreferredSize(new Dimension(150,18));
    l3.setForeground(color);
    l3.setVisible(false);
    l3.setFont(font1);
    panel.add(l3);

    t2=new JTextField(""+homeLongitude);
    t2.setPreferredSize(new Dimension(150,18));
    t2.setVisible(false);
    t2.setFont(font2);
    panel.add(t2);
    l4=new JLabel("ELEVATION");
    l4.setPreferredSize(new Dimension(150,18));
    l4.setForeground(color);
    l4.setVisible(false);
    l4.setFont(font1);
        panel.add(l4);
    t3=new JTextField(""+homeElevation);
    t3.setPreferredSize(new Dimension(150,18));
    t3.setVisible(false);
    t3.setFont(font2);
    panel.add(t3);
    l5=new JLabel("PITCH");
    l5.setPreferredSize(new Dimension(150,18));
    l5.setForeground(color);
    l5.setVisible(false);
    l5.setFont(font1);
    panel.add(l5);
    t4=new JTextField(""+homePitch);
    t4.setPreferredSize(new Dimension(150,18));
    t4.setVisible(false);
    t4.setFont(font2);
    panel.add(t4);
    l6=new JLabel("HEADING");
    l6.setPreferredSize(new Dimension(150,18));
    l6.setForeground(color);
    l6.setVisible(false);
    l6.setFont(font1);
    panel.add(l6);
    t5=new JTextField(""+homeHeading);
```

```
    t5.setPreferredSize(new Dimension(150,18));
    t5.setVisible(false);
    t5.setFont(font1);
    panel.add(t5);
}
private void setAreaPanel(Font font0,Font font1,Font font2,Color color){
    l7=new JLabel("MARKER | AREA");
    l7.setPreferredSize(new Dimension(150,18));
    l7.setForeground(color);
    l7.setVisible(false);
    l7.setFont(font0);
    panel.add(l7);
l17=new JLabel("HEIGHT");
l17.setPreferredSize(new Dimension(150,18));
  l17.setForeground(color);
  l17.setVisible(false);
l17.setFont(font1);
panel.add(l17);
t11=new JTextField(""+markerElevation);
t11.setPreferredSize(new Dimension(150,18));
t11.setVisible(false);
t11.setFont(font2);
panel.add(t11);
    l8=new JLabel("AREACOLOR");
    l8.setPreferredSize(new Dimension(150,18));
    l8.setForeground(color);
    l8.setVisible(false);
    l8.setFont(font1);
    panel.add(l8);
    t6=new JTextField("Color.RED");
    t6.setPreferredSize(new Dimension(150,18));
    t6.setVisible(false);
    t6.setFont(font2);
    panel.add(t6);
    l9=new JLabel("AREALINEWIDTH");
    l9.setPreferredSize(new Dimension(150,18));
    l9.setForeground(color);
    l9.setVisible(false);
    l9.setFont(font1);
    panel.add(l9);
    t7=new JTextField(""+polylineLineWidth);
    t7.setPreferredSize(new Dimension(150,18));
    t7.setVisible(false);
    t7.setFont(font2);
    panel.add(t7);
}
    private void setPlanePanel(Font font0,Font font1,Font font2,Color color){
    l11=new JLabel("PLANE");
    l11.setPreferredSize(new Dimension(150,18));
    l11.setForeground(color);
    l11.setVisible(false);
    l11.setFont(font0);
    panel.add(l11);
    l12=new JLabel("PLANEHEIGHT");
    l12.setPreferredSize(new Dimension(150,18));
    l12.setForeground(color);
    l12.setVisible(false);
    l12.setFont(font1);
    panel.add(l12);
    t8=new JTextField(""+planeElevation);
    t8.setPreferredSize(new Dimension(150,18));
    t8.setVisible(false);
    t8.setFont(font2);
    panel.add(t8);
    l13=new JLabel("PLANERADIUS");
    l13.setPreferredSize(new Dimension(150,18));
    l13.setForeground(color);
    l13.setVisible(false);
    l13.setFont(font2);
    panel.add(l13);
    t9=new JTextField(""+planeRadius);
    t9.setPreferredSize(new Dimension(150,18));
    t9.setVisible(false);
    t9.setFont(font2);
    panel.add(t9);
    }
```

```
private void setCreditPanel(Font font,Color color){
    l10=new JLabel("");
    l10.setPreferredSize(new Dimension(200,100));
    l10.setForeground(color);
    l10.setVisible(false);
    l10.setFont(font);
    panel.add(l10);
}
}
```