Building Java Applications in XAL

Thomas Pelaia II, Ph.D.

XAL Lecture Series

Lesson 3

July 8, 2008

Managed by UT-Battelle
for the Department of Energy

OAK RIDGE National Laboratory

XAL Lecture Series

1. **Java Primer**
2. **Tour of XAL**
3. **Building Java Applications in XAL**
4. **Ruby Primer**
5. **Scripting Applications in XAL**
6. **Channel Access in XAL**
7. **XAL Accelerator Hierarchy**

Constructing Applications

- Application Framework
  - foundation for modern XAL applications
- Bricks
  - graphical user interface construction
  - runtime for generating user interfaces
- Options for applications
  - Mix Bricks with Application Framework
    - recommended for new applications
  - Use Bricks alone
    - ideal for scripts with user interfaces
  - Use application framework alone
    - common among older applications

Application Framework

- Foundation for document based applications
- Consistent look and feel across applications
  - standard, familiar menu items
  - free automatic behaviors
- Rapid application development
- Broadly customizable
- Integrated online application help

Bricks-based Application Framework Components

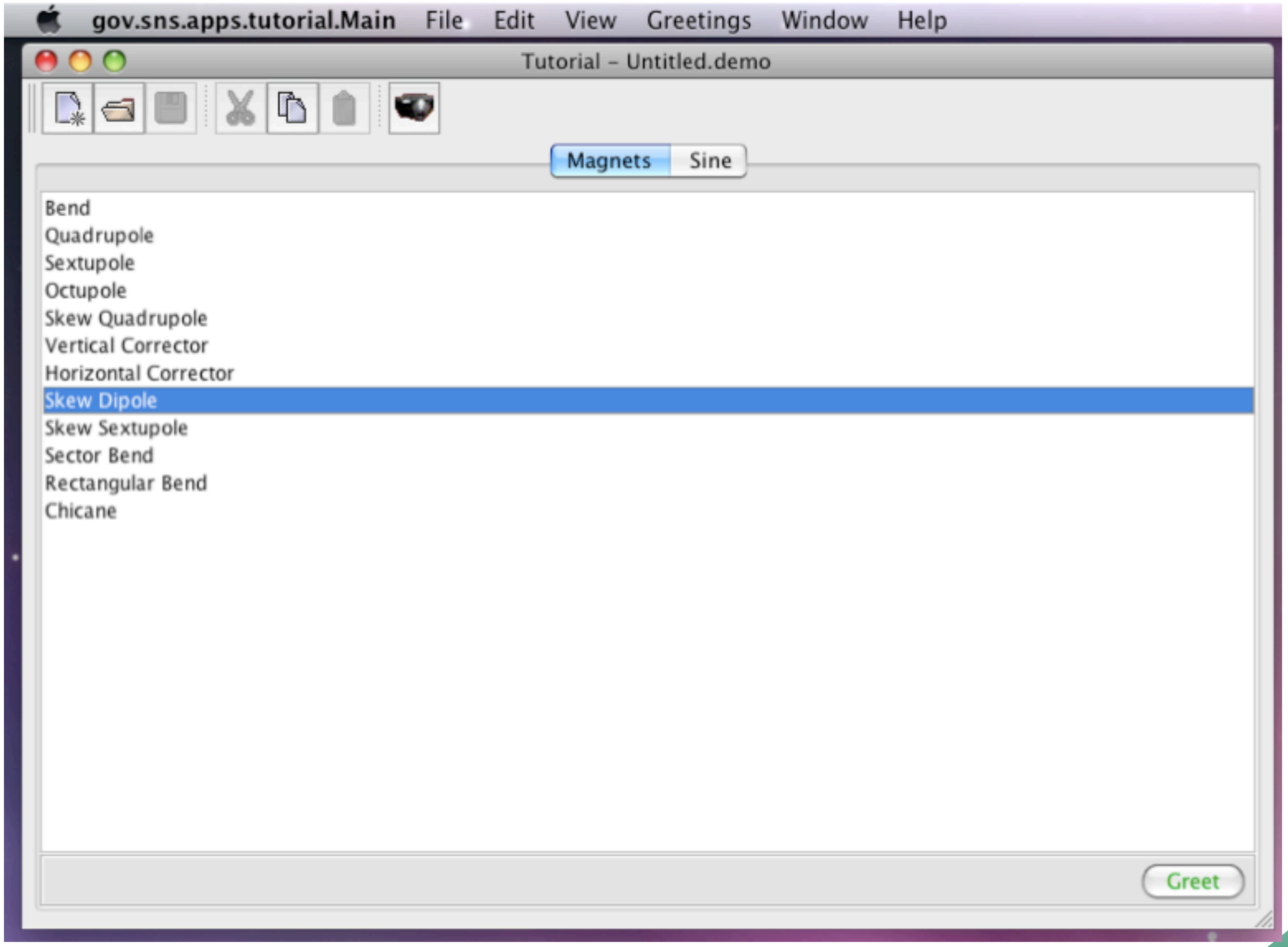
Base Class	Role of Custom Subclass
ApplicationAdaptor	Provide application specific information and hooks
XalDocument	Document specific attributes, data and hooks

Resource	Role
gui.bricks	XML file with definition of windows and dialogs
menudef.properties	Defines menus, menu items, toolbar and toolbar buttons
About.properties	Basic application information
Help.html	Online documentation

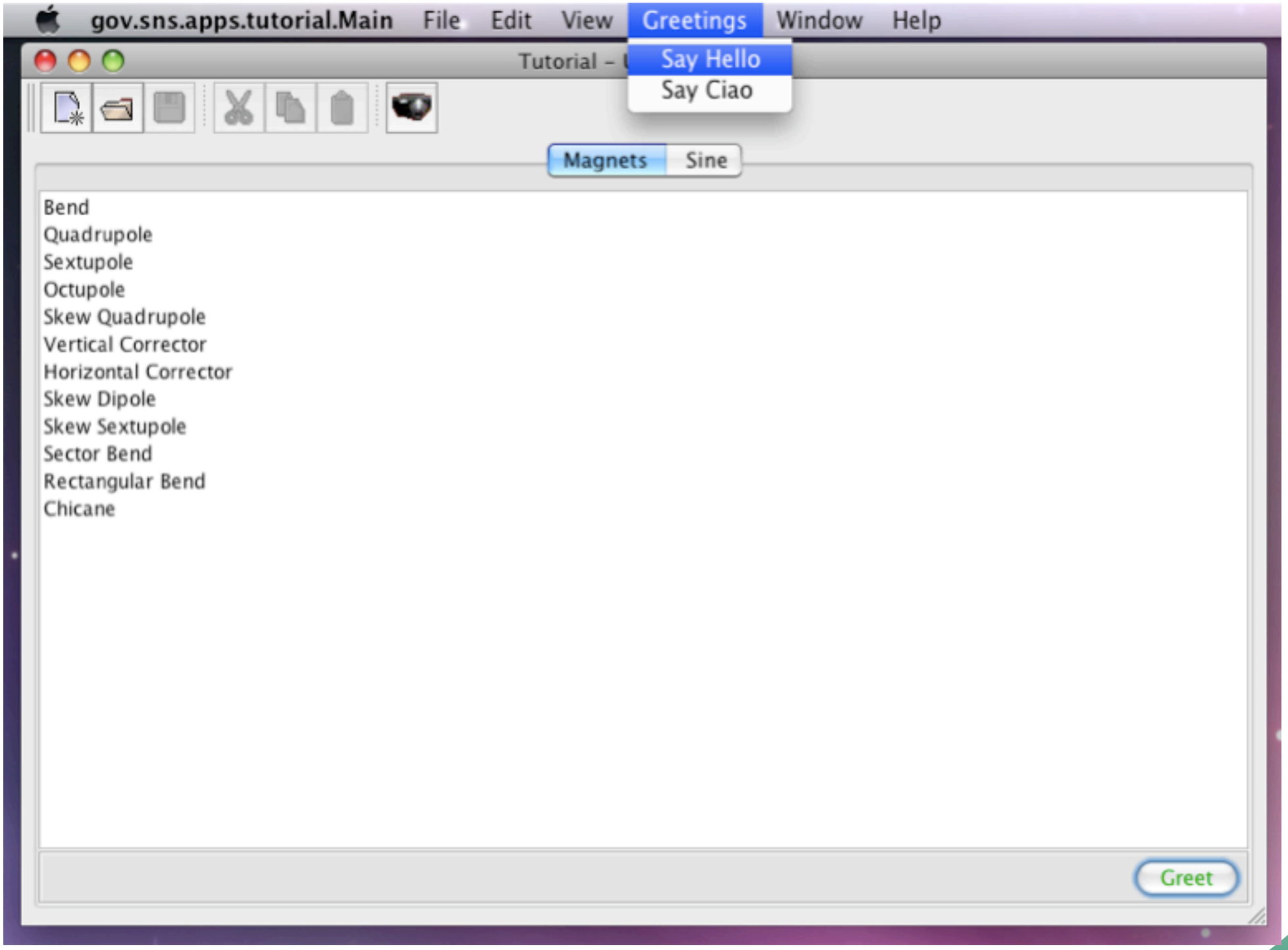
Hello World Tutorial

- Bricks based user interface
- XAL Application Framework foundation
- Simple application
- Best to start new applications by copying files from existing applications
  - Don’t copy directories under source code management (e.g. svn) since they contain hidden files

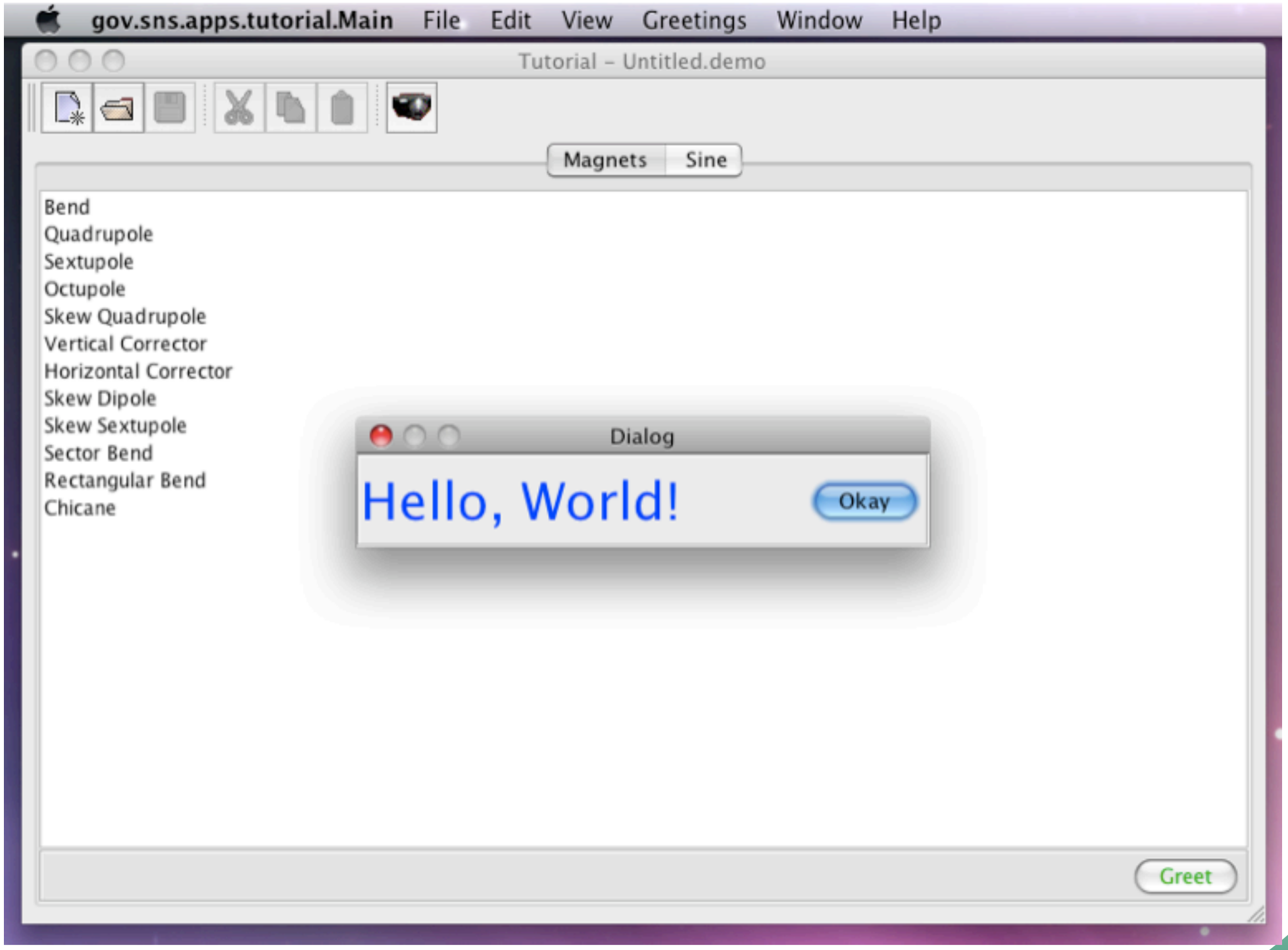
Tutorial Application - Sneak Peek



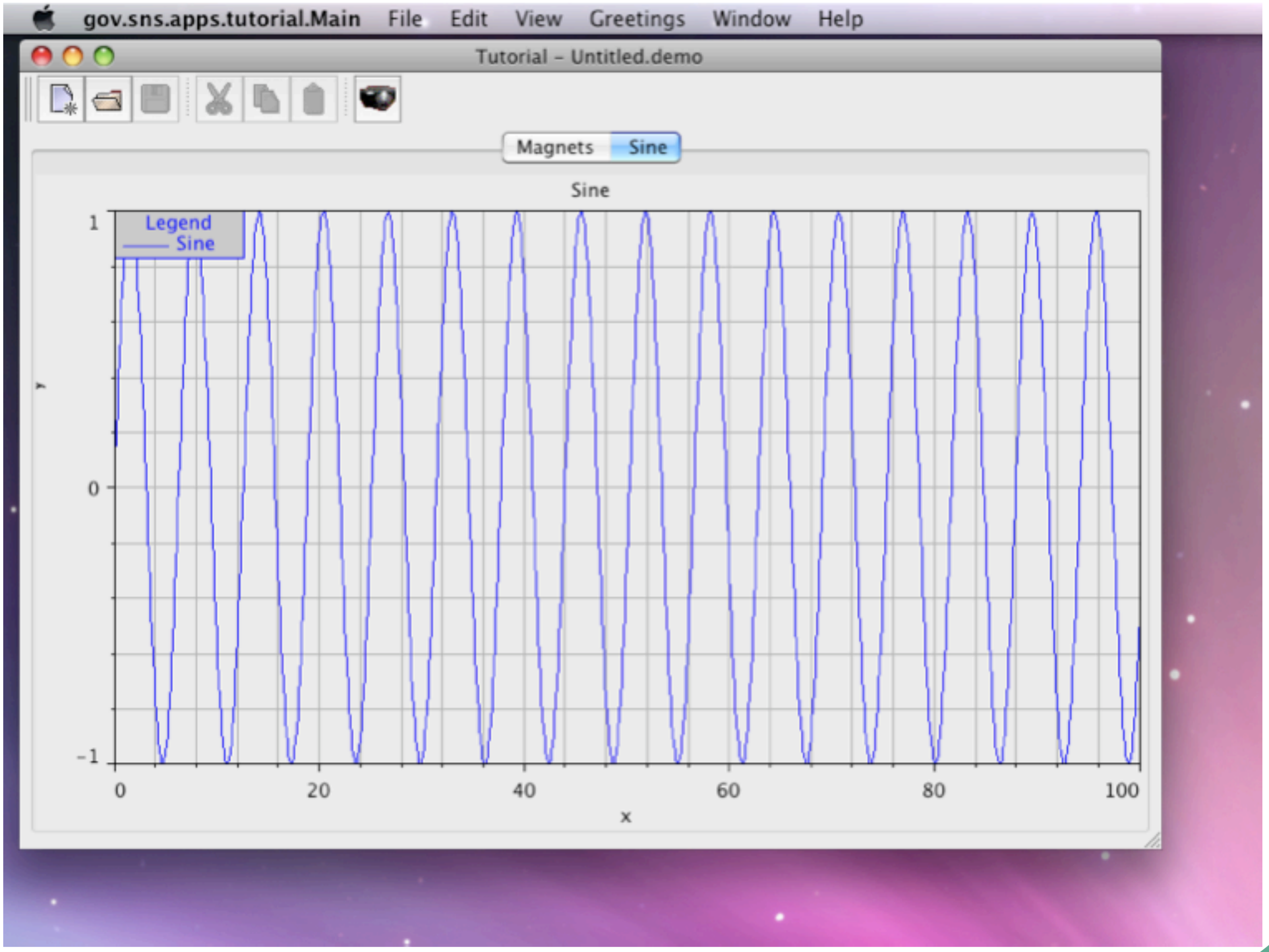
Tutorial Application - Sneak Peek



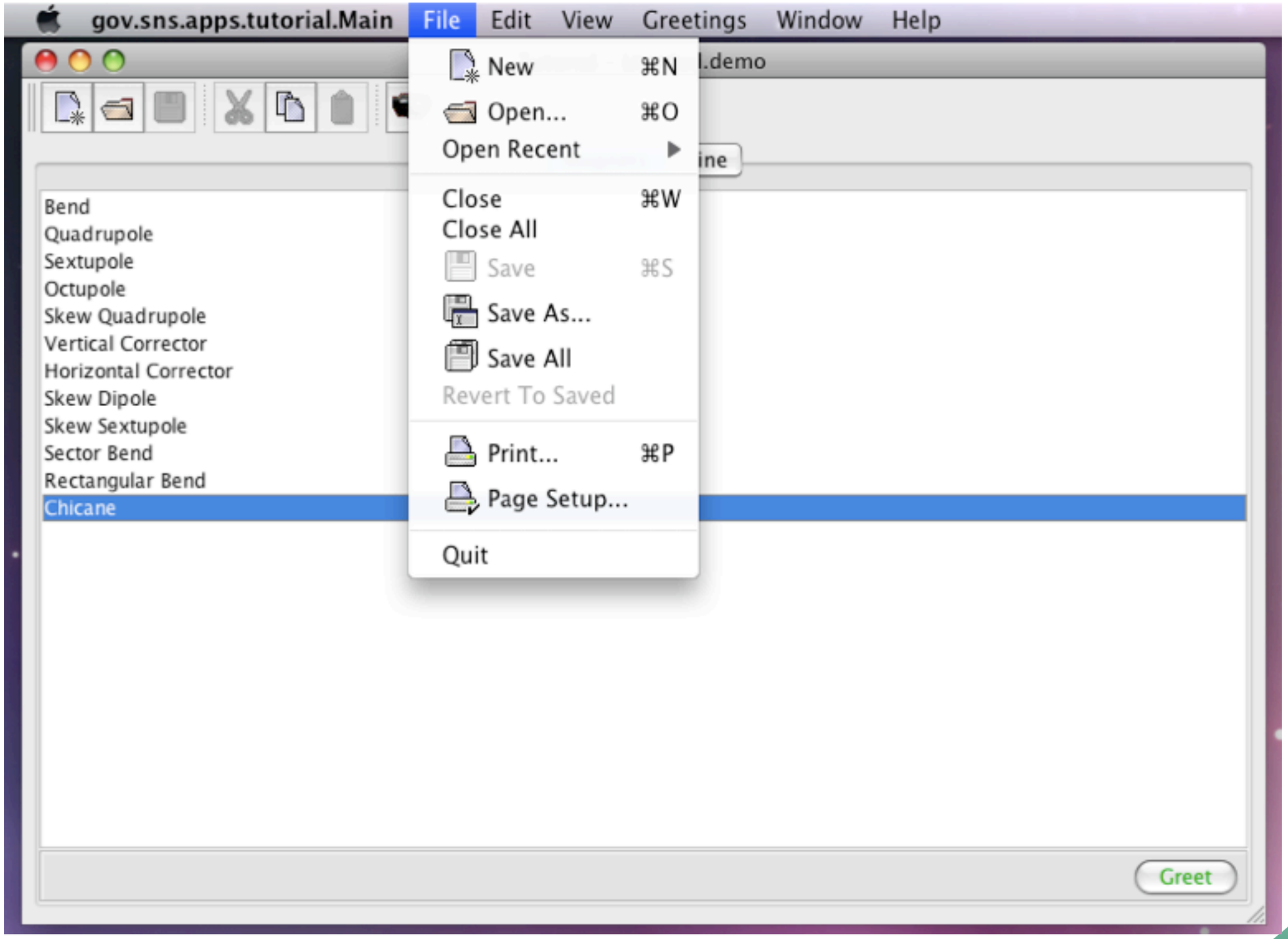
Tutorial Application - Sneak Peek



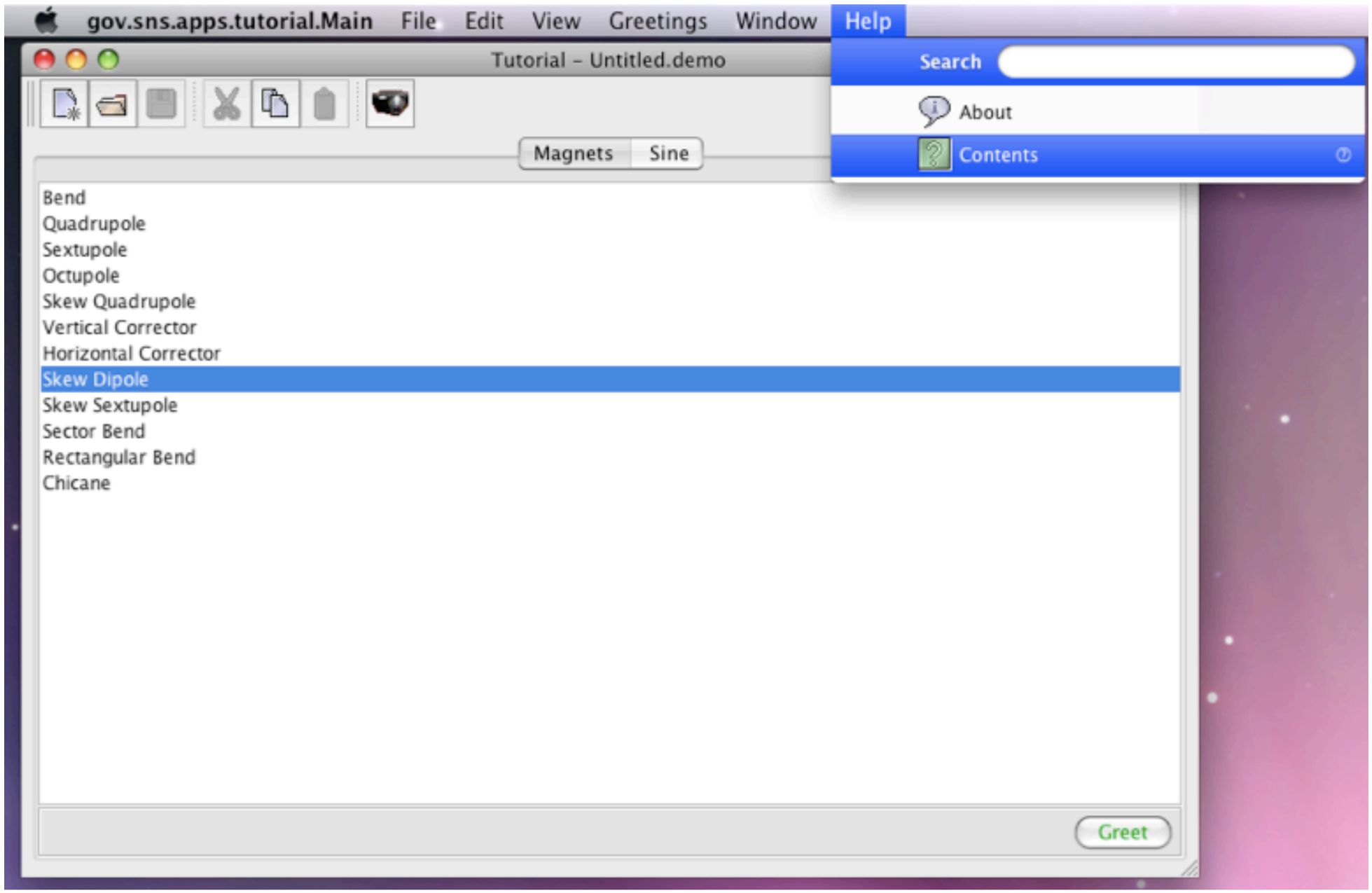
Tutorial Application - Sneak Peek



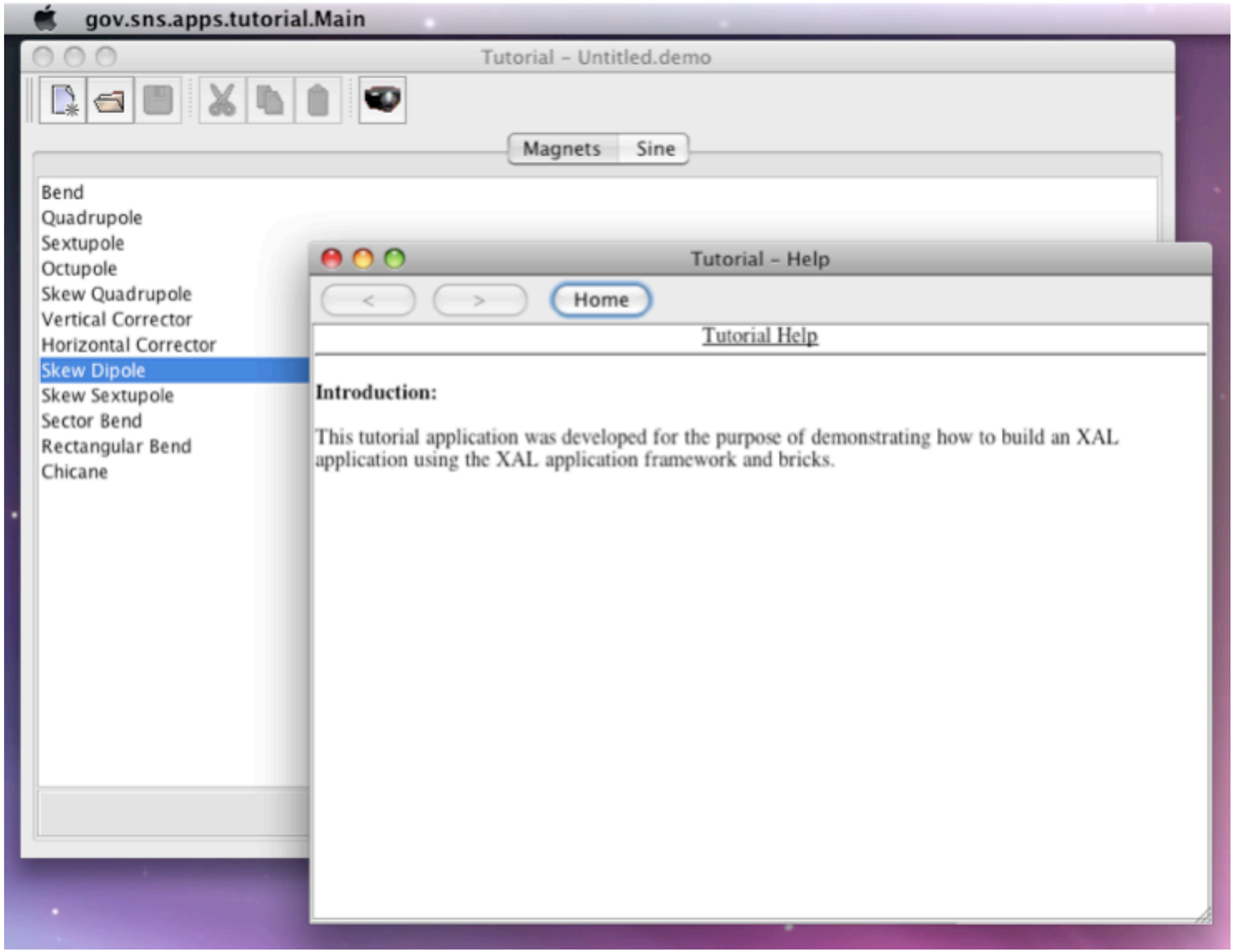
Tutorial Application - Sneak Peek



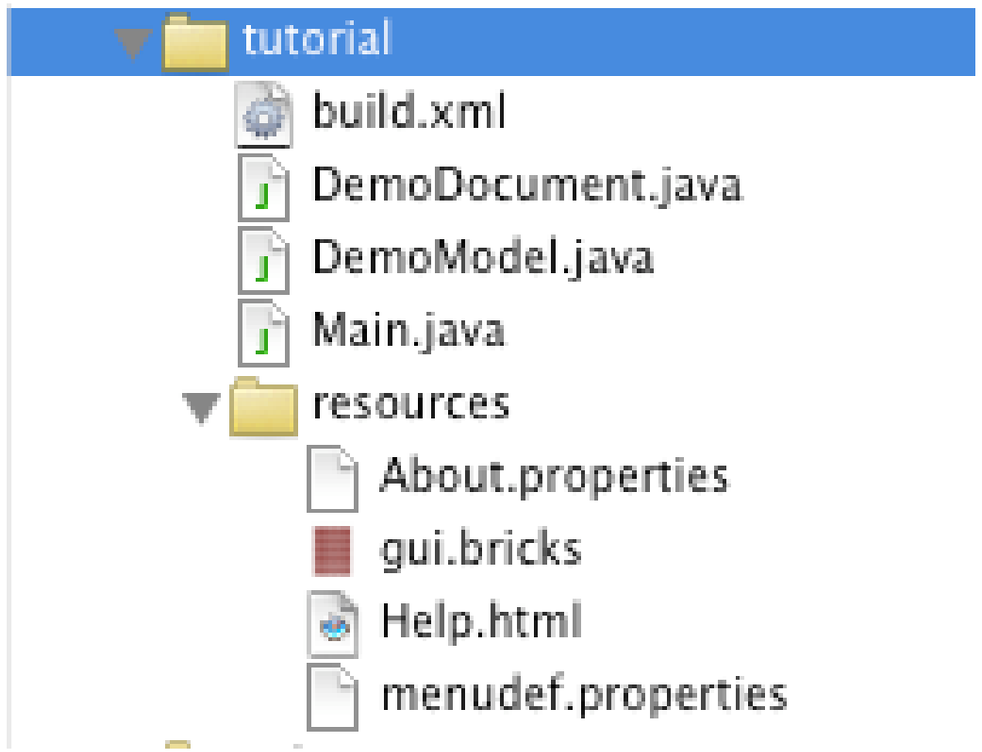
Tutorial Application - Sneak Peek



Tutorial Application - Sneak Peek



Tutorial File Hierarchy



Ant Build File
build.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!-- Build the application whose name matches that of the project. -->
<project name="tutorial" basedir="." default="all">
    <property environment="env"/>

  <import file="${env.XAL_HOME}/app_base.xml"/>
</project>
```

Application Information about.properties

```
name = XAL Tutorial
version = 1.0.0
date = July 1, 2008
authors = Tom Pelaia (SNS)
organization = Spallation Neutron Source
description = This application demonstrates how to create a simple application in XAL.
```

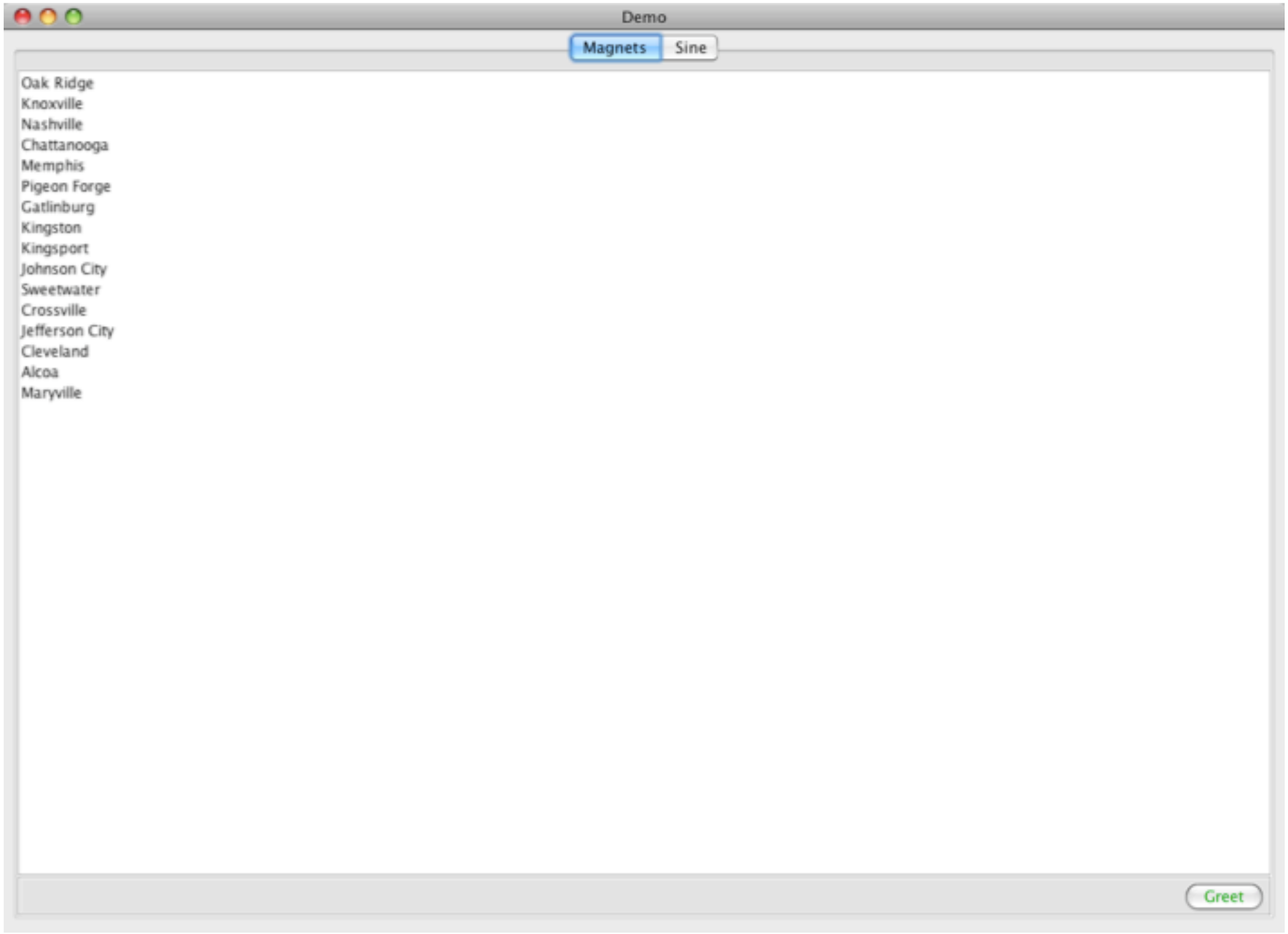
Online Help
Help.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<HTML>
<HEAD>
<TITLE>Tutorial Help</TITLE>
</HEAD>
<BODY>

<center><u>Tutorial Help</u></center>

<hr><br>
<a name="intro_ref" <b>Introduction:</b></a>
<p>
This tutorial application was developed for the purpose of demonstrating how to build an XAL application
using the XAL application framework and bricks.</p>

</BODY>
</HTML>
```

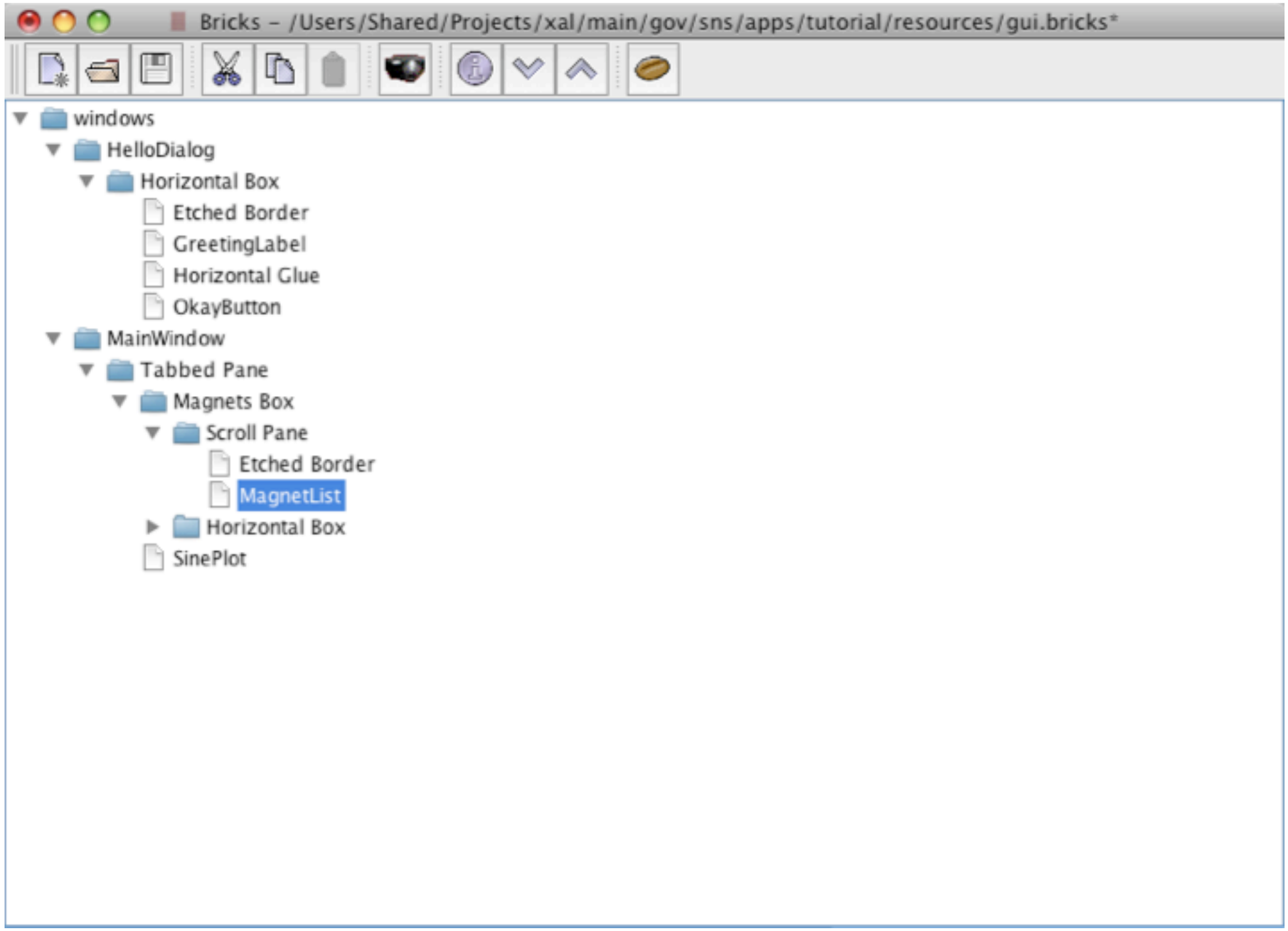
Menu Customization menudef.properties

```
# customize the menubar for the demo by adding a Greetings menu
menubar = file edit view greet window help


# define the custom menu labels
greet_label = Greetings


# define the custom menu items
greet_menu = say_hello say_ciao


# define the greeting menu item labels
say_hello_label = Say Hello
say_ciao_label = Say Ciao


# define actions
say_hello_action = say-hello
say_ciao_action = say-ciao
```

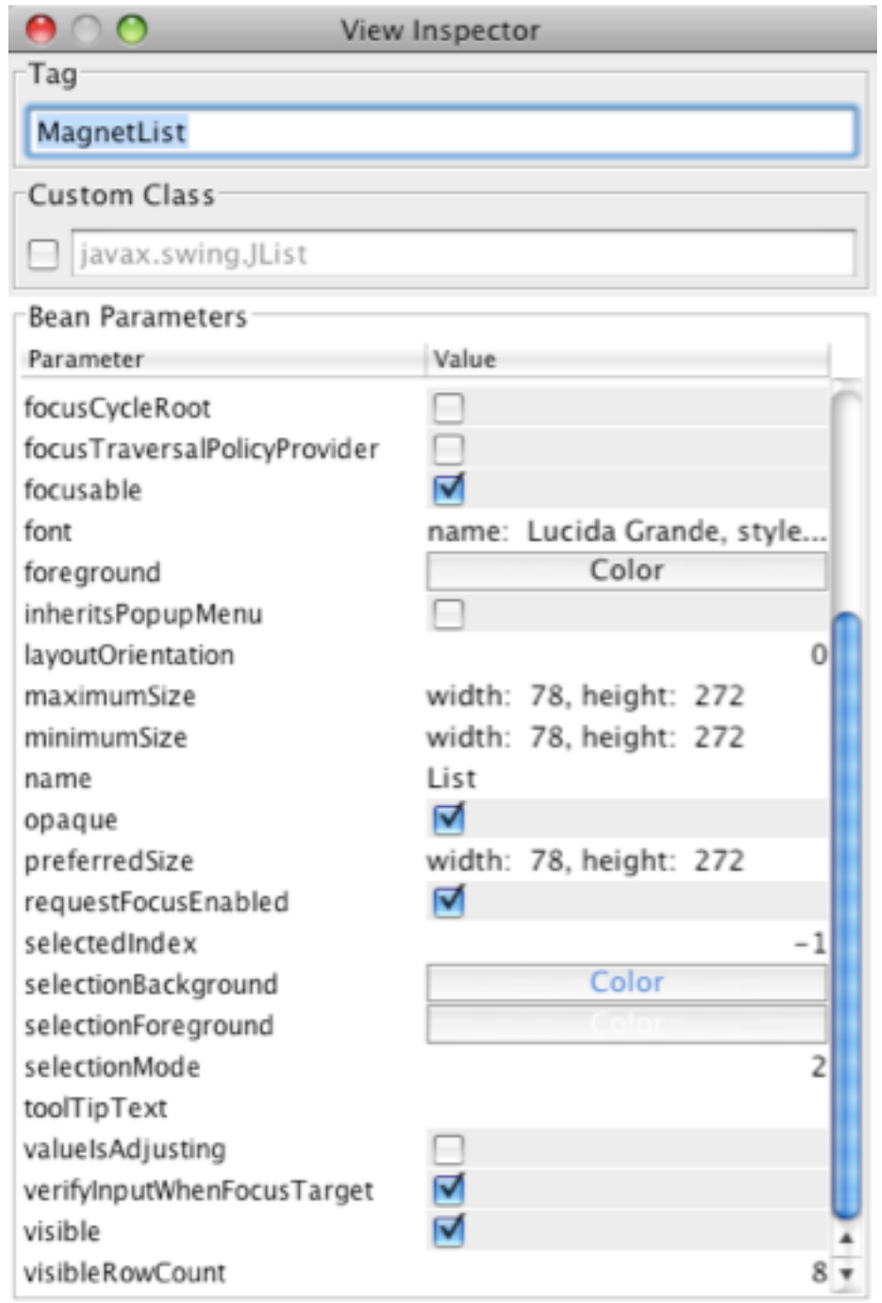
Bricks GUI Builder
gui.bricks



Bricks GUI Builder
gui.bricks



Bricks GUI Builder
gui.bricks



Bricks GUI Builder
gui.bricks

Graphical User Interface gui.bricks

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<BricksDocument date="Thu Jul 03 15:16:18 EDT 2008" version="1.0.0">
    <RootBrick>
        <ViewNode height="80" tag="HelloDialog" width="355">
            <ViewProxy type="javax.swing.JDialog"/>
            <BeanProperty name="focusableWindowState" value="true"/>
            <BeanProperty name="resizable" value="false"/>
            <BeanProperty name="modal" value="true"/>
            <BeanProperty fontName="Dialog" name="font" size="40" style="0"/>
            <ViewNode tag="Horizontal Box">
                <ViewProxy type="javax.swing.Box_Horizontal"/>
                <BorderNode tag="Etched Border">
                    <BorderProxy type="javax.swing.border.EtchedBorder"/>
                </BorderNode>
                <ViewNode tag="GreetingLabel">
                    <ViewProxy type="javax.swing.JLabel"/>
                    <BeanProperty name="text" value="Hello, World!"/>
                    <BeanProperty alpha="255" blue="255" green="51" name="foreground" red="0"/>
                    <BeanProperty fontName="Dialog" name="font" size="32" style="0"/>
                </ViewNode>
                <ViewNode tag="Horizontal Glue">
                    <ViewProxy type="javax.swing.Box_HorizontalGlue"/>
                </ViewNode>
                <ViewNode tag="OkayButton">
                    <ViewProxy type="javax.swing.JButton"/>
                    <BeanProperty name="text" value="Okay"/>
                </ViewNode>
                <BorderNode tag="Etched Border">
                    <BorderProxy type="javax.swing.border.EtchedBorder"/>
                </BorderNode>
            </ViewNode>
        </ViewNode>
        <ViewNode customBeanClass="gov.sns.application.DefaultXalWindow" height="547"
tag="MainWindow" width="772">
            <ViewProxy type="javax.swing.JFrame"/>
            <BeanProperty name="title" value="Demo"/>
```

```
                <ViewNode tag="Tabbed Pane">
                <ViewProxy type="javax.swing.JTabbedPane"/>
                <ViewNode tag="Magnets Box">
                    <ViewProxy type="javax.swing.Box_Vertical"/>
                    <BeanProperty name="name" value="Magnets"/>
                    <ViewNode tag="Scroll Pane">
                        <ViewProxy type="javax.swing.JScrollPane"/>
                        <BorderNode tag="Etched Border">
                            <BorderProxy type="javax.swing.border.EtchedBorder"/>
                        </BorderNode>
                        <ViewNode tag="MagnetList">
                            <ViewProxy type="javax.swing.JList"/>
                        </ViewNode>
                        <BorderNode tag="Etched Border">
                            <BorderProxy type="javax.swing.border.EtchedBorder"/>
                        </BorderNode>
                    </ViewNode>
                    <ViewNode tag="Horizontal Box">
                        <ViewProxy type="javax.swing.Box_Horizontal"/>
                        <BorderNode tag="Etched Border">
                            <BorderProxy type="javax.swing.border.EtchedBorder"/>
                        </BorderNode>
                        <ViewNode tag="Horizontal Glue">
                            <ViewProxy type="javax.swing.Box_HorizontalGlue"/>
                        </ViewNode>
                        <ViewNode tag="RunButton">
                            <ViewProxy type="javax.swing.JButton"/>
                            <BeanProperty name="toolTipText" value="Display a greeting dialog
box."/>
                            <BeanProperty name="text" value="Greet"/>
                            <BeanProperty alpha="255" blue="0" green="153" name="foreground"
red="0"/>
                        </ViewNode>
                        <BorderNode tag="Etched Border">
                            <BorderProxy type="javax.swing.border.EtchedBorder"/>
                        </BorderNode>
                    </ViewNode>
                </ViewNode>
                <ViewNode tag="SinePlot">
                    <ViewProxy type="gov.sns.tools.plot.FunctionGraphsJPanel"/>
                    <BeanProperty name="name" value="Sine"/>
                </ViewNode>
            </ViewNode>
        </ViewNode>
    </RootBrick>
</BricksDocument>
```

Application Adaptor Main.java

```java
/*
 * Main.java
 *
 * Created on July 3, 2008, 4:13 PM
 */

package gov.sns.apps.tutorial;

import javax.swing.*;
import java.net.*;
import java.util.logging.*;

import gov.sns.application.*;


/**
 * Main is a demo concrete subclass of ApplicationAdaptor.
 * @author  t6p
 */
public class Main extends ApplicationAdaptor {
    /**
     * Returns the text file suffixes of files this application can open.
     * @return Suffixes of readable files
     */
    public String[] readableDocumentTypes() {
        return new String[] {"demo"};
    }


    /**
     * Returns the text file suffixes of files this application can write.
     * @return Suffixes of writable files
     */
    public String[] writableDocumentTypes() {
        return new String[] {"demo"};
    }


    /**
     * Implement this method to return an instance of my custom document.
     * @return An instance of my custom document.
     */
    public XalDocument newEmptyDocument() {
        return new DemoDocument();
    }
```

```java
    /**
     * Implement this method to return an instance of my custom document
     * corresponding to the specified URL.
     * @param url The URL of the file to open.
     * @return An instance of my custom document.
     */
    public XalDocument newDocument( final java.net.URL url ) {
        return new DemoDocument( url );
    }


    /**
     * Specifies the name of my application.
     * @return Name of my application.
     */
    public String applicationName() {
        return "Tutorial";
    }


    /** Handle the application launch event and log a message. */
    public void applicationFinishedLaunching() {
        System.out.println( "Application has finished launching!" );
             Logger.getLogger("global").log( Level.INFO, "Application has finished
launching." );
    }


    /** The main method of the application. */
    static public void main( final String[] args ) {
        try {
            System.out.println( "Starting application..." );
                    Logger.getLogger("global").log( Level.INFO, "Starting
application..." );
            Application.launch( new Main() );
        }
        catch( Exception exception ) {
                    Logger.getLogger("global").log( Level.SEVERE, "Error starting
application.", exception );
            System.err.println( exception.getMessage() );
            exception.printStackTrace();
                    Application.displayApplicationError( "Launch Exception",
"Launch Exception", exception );
                    System.exit( -1 );
        }
    }
}
```

Application Adaptor - Class Header Main.java

```
/*
 * Main.java
 *
 * Created on July 3, 2008, 4:13 PM
 */

package gov.sns.apps.tutorial;

import javax.swing.*;
import java.net.*;
import java.util.logging.*;

import gov.sns.application.*;


/**
 * Main is a demo concrete subclass of ApplicationAdaptor.
 * @author  t6p
 */
public class Main extends ApplicationAdaptor {
```

Application Adaptor - Document Handling Main.java

```java
/**
 * Returns the text file suffixes of files this application can open.
 * @return Suffixes of readable files
 */
public String[] readableDocumentTypes() {
    return new String[] {"demo"};
}


/**
 * Returns the text file suffixes of files this application can write.
 * @return Suffixes of writable files
 */
public String[] writableDocumentTypes() {
    return new String[] {"demo"};
}


/**
 * Implement this method to return an instance of my custom document.
 * @return An instance of my custom document.
 */
public XalDocument newEmptyDocument() {
    return new DemoDocument();
}


/**
 * Implement this method to return an instance of my custom document
 * corresponding to the specified URL.
 * @param url The URL of the file to open.
 * @return An instance of my custom document.
 */
public XalDocument newDocument( final java.net.URL url ) {
    return new DemoDocument( url );
}
```

Application Adaptor - Application Handling Main.java

```
    /**
     * Specifies the name of my application.
     * @return Name of my application.
     */
    public String applicationName() {
        return "Tutorial";
    }


    /** Handle the application launch event and log a message. */
    public void applicationFinishedLaunching() {
        System.out.println( "Application has finished launching!" );
           Logger.getLogger("global").log( Level.INFO, "Application has finished launching." );
    }


    /** The main method of the application. */
    static public void main( final String[] args ) {
        try {
            System.out.println( "Starting application..." );
                Logger.getLogger("global").log( Level.INFO, "Starting application..." );
            Application.launch( new Main() );
        }
        catch( Exception exception ) {
                Logger.getLogger("global").log( Level.SEVERE, "Error starting application.", exception );
            System.err.println( exception.getMessage() );
            exception.printStackTrace();
                Application.displayApplicationError( "Launch Exception", "Launch Exception", exception );
                System.exit( -1 );
        }
    }
}
```

Model
DemoModel.java

```java
//
//  DemoModel.java
//  xal
//
//  Created by Tom Pelaia on 7/7/08.
//  Copyright 2008 Oak Ridge National Lab. All rights reserved.
//

package gov.sns.apps.tutorial;


/** Main document model */
public class DemoModel {
   /** magnets */
   final private String[] MAGNETS;

   /** Constructor */
   public DemoModel() {
         final String[] magnets = { "Bend", "Quadrupole", "Sextupole", "Octupole", "Skew Quadrupole", "Vertical Corrector",
"Horizontal Corrector", "Skew Dipole", "Skew Sextupole", "Sector Bend", "Rectangular Bend", "Chicane" };
         MAGNETS = magnets;
   }


   /** get the magnets */
   public String[] getMagnets() {
         return MAGNETS;
   }


   /** calculate the amplitude for the specified phase */
   public double calcAmplitude( final double phase ) {
         return Math.sin( phase );
   }
}
```

Document DemoDocument.java

```java
/*
 * DemoDocument.java
 * Created on July 3, 2008, 3:27 PM
 */

package gov.sns.apps.tutorial;

import java.net.*;
import java.io.*;
import java.awt.Color;
import javax.swing.*;
import javax.swing.JToggleButton.ToggleButtonModel;
import javax.swing.text.*;
import javax.swing.event.*;
import java.awt.event.*;
import java.util.logging.*;
import java.util.Vector;

import gov.sns.application.*;
import gov.sns.tools.bricks.WindowReference;
import gov.sns.tools.plot.*;


/**
 * DemoDocument is a custom XalDocument for this application.
 * @author  t6p
 */
public class DemoDocument extends XalDocument {
      /** main model */
      final DemoModel MODEL;


    /** Create a new empty document */
    public DemoDocument() {
        this( null );
    }


    /**
     * Create a new document loaded from the URL file
     * @param url The URL of the file to load into the new document.
     */
    public DemoDocument( final URL url ) {
        setSource( url );

                MODEL = new DemoModel();
    }

    /**
     * Save the document to the specified URL.
     * @param url The URL to which the document should be saved.
     */
    public void saveDocumentAs( final URL url ) {
    }


    /**
     * Make a main window by instantiating the my custom window.  Set the
text
     * pane to use the textDocument variable as its document.
     */
    public void makeMainWindow() {
                final WindowReference windowReference =
getDefaultWindowReference( "MainWindow", this );
        mainWindow = (XalWindow)windowReference.getWindow();

                final JList magnetList =
(JList)windowReference.getView( "MagnetList" );
                magnetList.setListData( MODEL.getMagnets() );

                final JButton runButton =
(JButton)windowReference.getView( "RunButton" );
                runButton.addActionListener( new ActionListener() {
                        public void actionPerformed( final ActionEvent
event ) {
                                displayGreeting();
                        }
                });

                final FunctionGraphsJPanel plot =
(FunctionGraphsJPanel)windowReference.getView( "SinePlot" );
                makeSinePlot( plot );
      }


      /** make the plot of the sine function from zero to 100 */
      protected void makeSinePlot( final FunctionGraphsJPanel plot ) {
                final BasicGraphData graphData = new BasicGraphData();
                graphData.setGraphColor( Color.BLUE );
                graphData.setGraphProperty( plot.getLegendKeyString(),
"Sine" );
                graphData.setDrawPointsOn( false );
                for ( double phase = 0.0 ; phase < 100.0 ; phase+=0.1 ) {
                        graphData.addPoint( phase,
MODEL.calcAmplitude( phase ) );
                }

                plot.addGraphData( graphData );
      }


      /** Display the default greeting */
      protected void displayGreeting() {
                displayGreeting( null );
      }

      /** Display the greeting dialog with the specified greeting */
      protected void displayGreeting( final String greeting ) {
                final WindowReference greetingDialogRef =
getDefaultWindowReference( "HelloDialog", mainWindow );
                final JDialog dialog =
(JDialog)greetingDialogRef.getWindow();

                final JButton okayButton =
(JButton)greetingDialogRef.getView( "OkayButton" );
                dialog.getRootPane().setDefaultButton( okayButton );
                okayButton.addActionListener( new ActionListener() {
                        public void actionPerformed( final ActionEvent
event ) {
                                dialog.setVisible( false );
                        }
                });

                if ( greeting != null ) {
                        final JLabel greetingLabel =
(JLabel)greetingDialogRef.getView( "GreetingLabel" );
                        greetingLabel.setText( greeting );
                }

                dialog.setLocationRelativeTo( mainWindow );
                dialog.setVisible( true );
      }


    /**
     * Implement the custom commands for the menu items.
     * @param commander The commander that manages commands.
     */
    protected void customizeCommands( final Commander commander ) {
        final Action sayHelloAction = new AbstractAction( "say-hello" ) {
            public void actionPerformed( final ActionEvent event ) {
                                displayGreeting( "Hello, Everyone!" );
            }
        };
        commander.registerAction( sayHelloAction );

        final Action sayCiaoAction = new AbstractAction( "say-ciao" ) {
            public void actionPerformed( final ActionEvent event ) {
                                displayGreeting( "Ciao!" );
            }
        };
        commander.registerAction( sayCiaoAction );
    }
}
```

Document - Header DemoDocument.java

```
/*
 * DemoDocument.java
 * Created on July 3, 2008, 3:27 PM
 */

package gov.sns.apps.tutorial;

import java.net.*;
import java.io.*;
import java.awt.Color;
import javax.swing.*;
import javax.swing.JToggleButton.ToggleButtonModel;
import javax.swing.text.*;
import javax.swing.event.*;
import java.awt.event.*;
import java.util.logging.*;
import java.util.Vector;

import gov.sns.application.*;
import gov.sns.tools.bricks.WindowReference;
import gov.sns.tools.plot.*;


/**
 * DemoDocument is a custom XalDocument for this application.
 * @author  t6p
 */
public class DemoDocument extends XalDocument {
```

Document - Instance Initialization DemoDocument.java

```
/** main model */
final DemoModel MODEL;


 /** Create a new empty document */
 public DemoDocument() {
     this( null );
 }


 /**
  * Create a new document loaded from the URL file
  * @param url The URL of the file to load into the new document.
  */
public DemoDocument( final URL url ) {
    setSource( url );

     MODEL = new DemoModel();
}


/**
 * Save the document to the specified URL.
 * @param url The URL to which the document should be saved.
 */
public void saveDocumentAs( final URL url ) {
}
```

Document - Main View Bindings DemoDocument.java

```
/**
 * Make a main window by instantiating the my custom window.  Set the text
 * pane to use the textDocument variable as its document.
 */
public void makeMainWindow() {
        final WindowReference windowReference = getDefaultWindowReference( "MainWindow", this );
        mainWindow = (XalWindow)windowReference.getWindow();

        final JList magnetList = (JList)windowReference.getView( "MagnetList" );
        magnetList.setListData( MODEL.getMagnets() );

        final JButton runButton = (JButton)windowReference.getView( "RunButton" );
        runButton.addActionListener( new ActionListener() {
              public void actionPerformed( final ActionEvent event ) {
                    displayGreeting();
              }
        });

        final FunctionGraphsJPanel plot = (FunctionGraphsJPanel)windowReference.getView( "SinePlot" );
        makeSinePlot( plot );
}


/** make the plot of the sine function from zero to 100 */
protected void makeSinePlot( final FunctionGraphsJPanel plot ) {
        final BasicGraphData graphData = new BasicGraphData();
        graphData.setGraphColor( Color.BLUE );
        graphData.setGraphProperty( plot.getLegendKeyString(), "Sine" );
        graphData.setDrawPointsOn( false );
        for ( double phase = 0.0 ; phase < 100.0 ; phase+=0.1 ) {
              graphData.addPoint( phase, MODEL.calcAmplitude( phase ) );
        }

        plot.addGraphData( graphData );
}


/** Display the default greeting */
protected void displayGreeting() {
        displayGreeting( null );
}
```

Document - Dialog Display DemoDocument.java

```
/** Display the greeting dialog with the specified greeting */
   protected void displayGreeting( final String greeting ) {
         final WindowReference greetingDialogRef = getDefaultWindowReference( "HelloDialog", mainWindow );
         final JDialog dialog = (JDialog)greetingDialogRef.getWindow();

         final JButton okayButton = (JButton)greetingDialogRef.getView( "OkayButton" );
         dialog.getRootPane().setDefaultButton( okayButton );
         okayButton.addActionListener( new ActionListener() {
             public void actionPerformed( final ActionEvent event ) {
                 dialog.setVisible( false );
             }
         });

         if ( greeting != null ) {
             final JLabel greetingLabel = (JLabel)greetingDialogRef.getView( "GreetingLabel" );
             greetingLabel.setText( greeting );
         }

         dialog.setLocationRelativeTo( mainWindow );
         dialog.setVisible( true );
   }
```

Document - Menu Customization DemoDocument.java

```
    /**
     * Implement the custom commands for the menu items.
     * @param commander The commander that manages commands.
     */
     protected void customizeCommands( final Commander commander ) {
         final Action sayHelloAction = new AbstractAction( "say-hello" ) {
             public void actionPerformed( final ActionEvent event ) {
                  displayGreeting( "Hello, Everyone!" );
             }
         };
         commander.registerAction( sayHelloAction );

         final Action sayCiaoAction = new AbstractAction( "say-ciao" ) {
             public void actionPerformed( final ActionEvent event ) {
                  displayGreeting( "Ciao!" );
             }
         };
         commander.registerAction( sayCiaoAction );
     }
}
```

Compile an Application (using Ant)

1. **Change directory to the top of the application's source code tree**
   - `cd ${XAL_HOME}/gov/sns/apps/tutorial`
2. **Type the Ant build command**
   - `ant`

Application Packaging

- Applications are packaged with relative references to the *xal.jar* and *ext.jar* files
- They are installed in the *build/jar/apps* directory under the XAL root
- To run an application named "tutorial" type:

```
java -jar ${XAL_HOME}/build/apps/tutorial.jar
```

Resources

- XAL Project
  - http://sourceforge.net/projects/xaldev
  - http://www.ornl.gov/~t6p/Main/XAL.html