Visualizer's Language

François Coupleux
coupleux@eleve.emn.fr

July 25, 1996

Contents

1 Introduction

Visualizer is a tutorial viewer. It is not the tutorial itself. Consequently, you have to describe the tutorial you want to design in order to make it understandable by *Visualizer*. This is the role of the tutorial description language.

2 Windows

2.1 Anatomy of a window

Visualizer has three types of windows:

- menus,
- plotting systems,
- customizable windows.

Although these windows are different, they have the same anatomy. They all have:

- a title,
- a content,
- a path showing the location of the window in the tutorial (bottom of the window),
- a button to go to the previous window (button '`Go Back`').

Moreover, if the content of a window is too high to be displayed, the window is divided into several pages. In this case, two new buttons are displayed:

- button '`Previous`',
- button '`Next`'.

They are used to browse the pages.

The anatomy of a window is represented in Fig. 1.

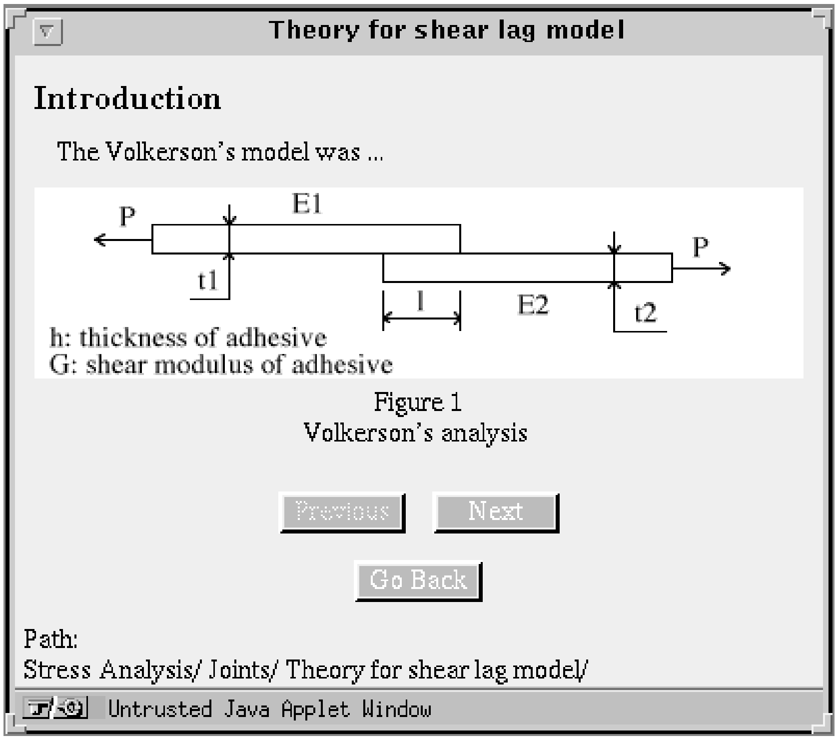


Figure 1

2.2 Menus

A menu is a window containing a set of buttons. When the user clicks on one button, *Visualizer* displays the window associated with the selected button. An example of menu is shown below.

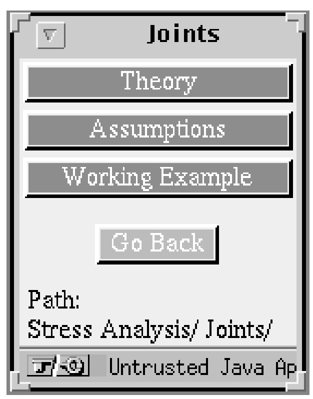


Figure 2

2.3 Plotting systems

A plotting system is an interface that allows the user to plot curves. It is composed of three different windows:

- a graph viewer (window displaying the graph),
- a window showing the legend of the current graph,
- a control window (window used to enter the values of parameters).

The screen shot below gives an example of a plotting system.

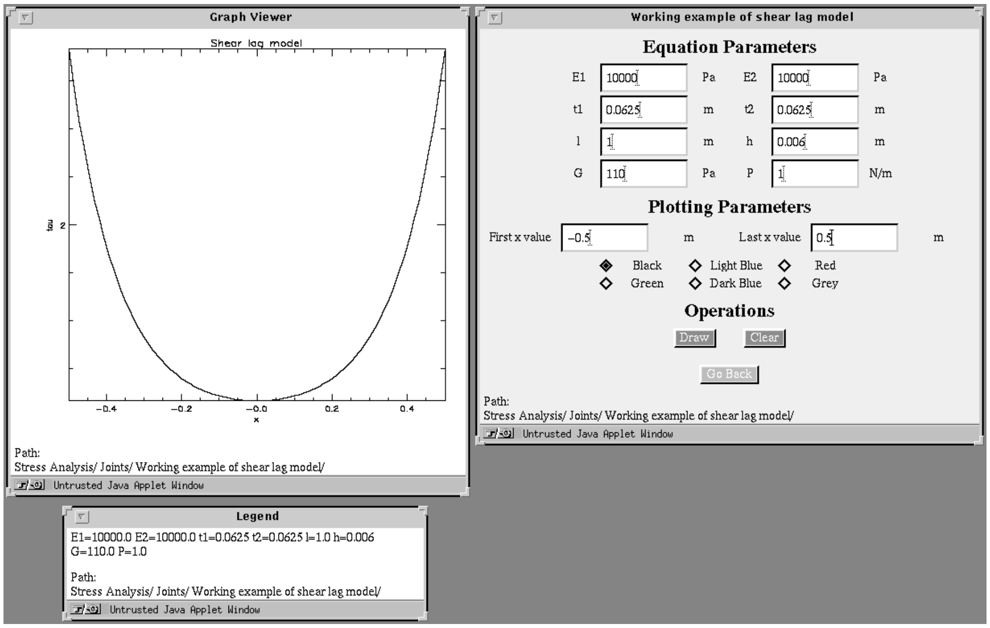


Figure 3

2.4 Customizable windows

A customizable window is a window used to display the content of the tutorial. *Visualizer* has a set of four standard elements that can be displayed in this type of window:

- titles,
- texts,
- navigators,
- pictures.

We will see each of these element in details later.

An example of a customizable window is given in figure 4.

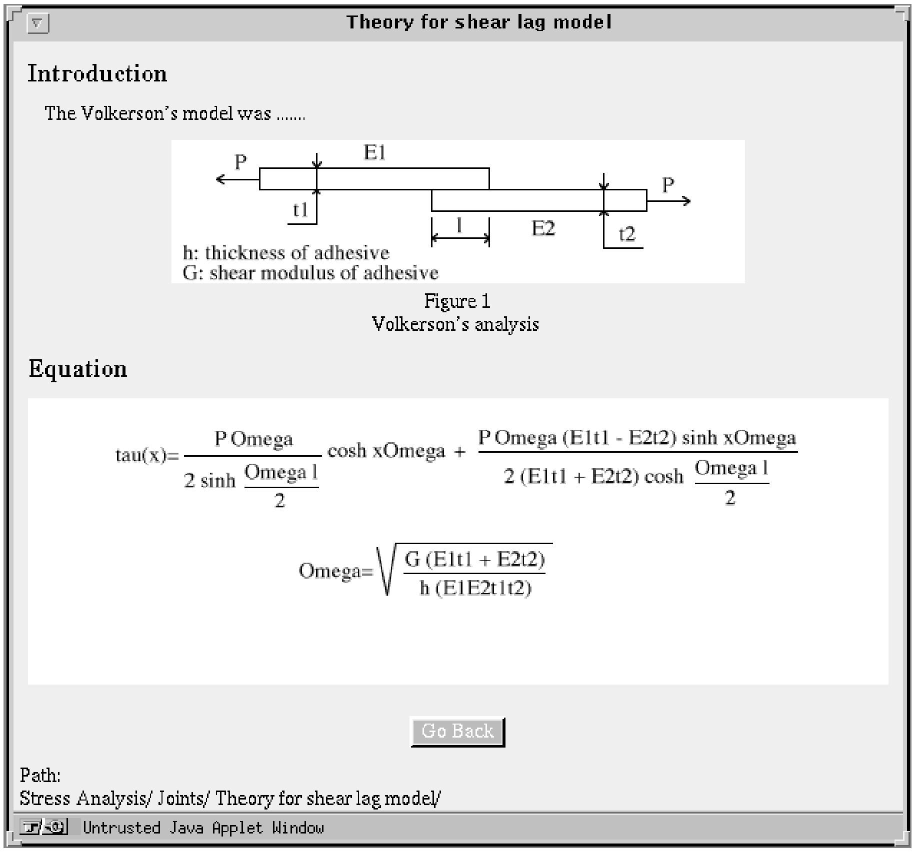


Figure 4

3 How can I program *Visualizer* ?

3.1 Generation of the description file

In order to display windows, *Visualizer* reads the description of the tutorial in the file *tutorialdescription.dat*. This file is in the applet's directory (*/home3/httpd/htdocs/javaprj/viscprj/visualizer/applet*). It is important to know that *Visualizer* does not check if the content of this file is correct. This task is done by a C program called *tutorialchecker*.

To describe a tutorial, you have to edit a description file in */home3/httpd/ htdocs/javaprj/viscprj/visualizer/tutorialchecker*. This file must not be called *tutorialdescription.dat*. Then, you have to run *tutorialchecker* by typing:

`tutorialchecker` *descriptionFile*

The file *descriptionFile* is checked and if it is correct the file *tutorialdescription.dat* is generated.

You have to stop servers if they are running. To do so, type:

```
ps -A | grep java
```

Then, type:

```
kill -9 process_number
```

For example, imagine we are in the following situation:

```
wave% ps -A | grep java
 17774 pts/0    0:05 java
 17779 pts/0    0:04 java
```

To kill the servers, you have to type:

```
kill -9 17774
kill -9 17779
```

The file *tutorialdescription.dat* must be moved into the applet's directory (*/home3/httpd/htdocs/javaprj/viscprj/visualizer/applet*). You can now restart the servers. You have to go to */home3/httpd/htdocs/javaprj/viscprj/visualizer/applet* and type:

```
java TutorialDescriptionServer &
java PVWaveAccessServer &
```

A tutorial description can be divided into three main steps:

1. declaration of the section setting the default values of attributes,
2. declaration of the main menu,
3. declaration of windows.

3.2 Setting default values

The aim of this section is to set several attributes which describe the general look of windows and elements. This section is not necessary. If it is not declared *Visualizer* will use its own default attributes. However if it is declared, it must be the first one of the description.

The declaration of this section is given below. You can refer to section 5 for an example of this declaration.

```
defaultvalues
attribute_1 arg_1 arg_2 ...
attribute_2 arg_1 arg_2 ...
...
enddefaultvalues
```

The table below gives the list of all the attributes which can be set here.

Attribute	Default value	Meaning
`backgroundcolor`	white	window's background color
`foregroundcolor`	black	color of the path
`titlecolor`	black	titles' color
`textcolor`	black	texts' color
`buttoncolor`	white	buttons' color
`labelcolor`	black	color of button's labels
`specialbuttoncolor`	white	color of special buttons: ('`Previous`', '`Next`' and '`Go Back`')
`speciallabelcolor`	black	color of special buttons' labels: ('`Previous`', '`Next`' and '`Go Back`')
`textfieldcolor`	white	text fields' color
`textwidth`	400	texts' width
`maxcontentheight`	700	maximal height of the content of a window
`xgap`	0	horizontal gap between the edges of the window and the edges of the widest element
`ygap`	0	vertical gap between elements

It is not necessary to set all these attributes. You just have to set the ones you want to change.

There are two types of attributes:

- colors,
- dimensions.

For *Visualizer*, a color is defined thanks to its red, green and blue shades. The syntax to set a color is the following:

```
color red green blue
```

For example if you want to have blue titles, you just have to type:

```
titlecolor 0 0 255
```

The three shades must be positive integers less than 256.

The second type of attribute is dimension in pixels. To be set, it just requires an integer. For example if you want to set the `xgap` attribute to 10, you can type:

```
xgap 10
```

3.3 Creating the main menu

The main menu is the menu located in the web browser. This is the first menu of the tutorial.

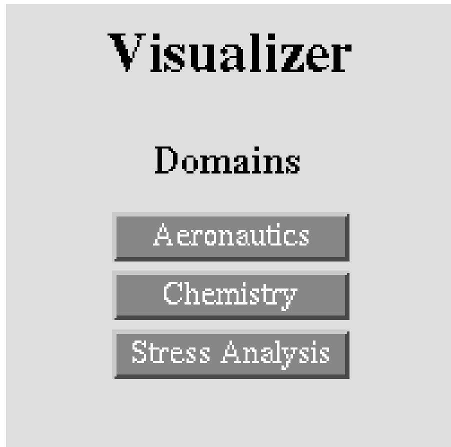


Figure 5

This section must be declared. It has to be:

- the first one if the section `defaultvalues` is not set,
- the second one if the section `defaultvalues` is set.

Here is the syntax to define the main menu:

```
mainmenu
attribute_1 arg_1 arg_2 ...
attribute_2 arg_1 arg_2 ...
...

declaration of a title (optional)

{ Button 1 & window_1 }
{ Button 2 & window_2 }
...
endmainmenu
```

An exemple of main menu is given in section 5.

You can set three attributes in this section:

Attribute	Meaning
`backgroundcolor`	window's background color
`buttoncolor`	buttons' color
`labelcolor`	color of button's labels

You are not required to set them. But if you decide to set one of these, the new value will be used for the main menu. But default values are not modified.

However, the main menu must contain an item. To define an item of the menu, you just have to type the following line:

```
{ title of the button & name_of_associated_window }
```

The first term of this expression is the title of the button. It can contain several words. The second term is the name of the next window to display if the user clicks on this button. It must be a single word. This name is the id of the window.

For example the following declaration

```
{ Stress Analysis & stressanalysis }
```

means that the label of the button is `Stress Analysis`. Moreover, when the user will click on this button, *Visualizer* will display the window called `stressanalysis`.

It is also possible to put a title in the main menu. The syntax to declare a title is described in section 4.1.

3.4 Declaring windows

The example given in section 5 gives you an illustration of the way to declare the three types of windows (menus, plotting systems, customizable windows).

3.4.1 Name and title

Every kind of window must have a name and a title. The title will be displayed in the title bar of the window. The name of the window is not displayed. These two attributes are set thanks to two keywords: `name` (name of the window) and `windowtitle` (title of the window). The title of a window can contain several words.

3.4.2 Menu

The simpliest window to declare is a menu. Here is the syntax to describe a menu:

```
menu
name name_of_window
windowtitle title of the window

attribute_1 arg_1 arg_2 ...
attribute_2 arg_1 arg_2 ...
...
```

```
{ Button 1 & window_1 }
{ Button 2 & window_2 }
...
endmenu
```

The attributes that can be set here are:

Attribute	Meaning
`backgroundcolor`	window's background color
`foregroundcolor`	color of the path
`buttoncolor`	buttons' color
`labelcolor`	color of button's labels
`maxcontentheight`	maximal height of the content of a window

The declaration of buttons is explained in section 3.3 (Creating the main menu).

3.4.3 Plotting system

A plotting system is used to plot curves. The syntax to declare it is described below.

```
plottingsystem
name name_of_window
windowtitle title of the window
xvariable x_variable_name
yvariable y_variable_name

attribute_1 arg_1 arg_2 ...
attribute_2 arg_1 arg_2 ...
...

declaration of equations

declaration of parameters

endplottingsystem
```

Two attributes are required:

Attribute	Meaning
`xvariable`	X variable's name
`yvariable`	Y variable's name

The value of these two attributes must be a single word.

The optional attributes that can be set in this section are given in the table below.

Attribute	Meaning
`backgroundcolor`	window's background color
`foregroundcolor`	color of the path
`titlecolor`	titles' color
`textcolor`	texts' color
`buttoncolor`	buttons' color
`labelcolor`	color of button's labels
`textfieldcolor`	text fields' color
`maxcontentheight`	maximal height of the content of a window
`graphtitle`	title of the graph
`xtitle`	title used for the X axis
`ytitle`	title used for the Y axis
`xvariableunit`	unit used for the X coordinates

It is necessary to declare two new subsections:

- `equations` (declaration of the equations that will be evaluated to plot curves)
- `attributes` (declaration of the parameters of equations)

The subsection `equations` must be declared before the subsection `attributes`.

The way to declare equations is very simple.

```
equations
{ equation 1 }
{ equation 2 }
...
endequations
```

It is important to know that equations are not checked by *tutorialchecker*. Consequently, you have to verify the form of these equations.

The syntax to declare parameters is similar to the syntax used to declare buttons.

```
parameters
{ parameter_1 & unit_parameter_1 }
{ parameter_2 & unit_parameter_2 }
...
endattributes
```

The first term is the name of the parameter and the second term is the unit corresponding to the parameter. Both must be a single word.

3.4.4 Customizable window

Here is the way to declare a customizable window:

```
window
name name_of_window
windowtitle title of the window

attribute_1 arg_1 arg_2 ...
attribute_2 arg_1 arg_2 ...
...

declaration of element 1
declaration of element 2
...
endwindow
```

The optional attributes that can be set here are:

Attribute	Meaning
`backgroundcolor`	window's background color
`foregroundcolor`	color of the path
`maxcontentheight`	maximal height of the content of a window
`xgap`	horizontal gap between the edges of the window and the edges of the widest element
`ygap`	vertical gap between elements

The description of elements will be explained in the next section.

4 Standard elements

4.1 Title

Here is an example of title:

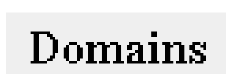


Figure 6

The declaration of a title is very simple:

```
title
attribute_1 arg_1 arg_2 ...
attribute_2 arg_1 arg_2 ...
```

```
...

{ title }
endtitle
```

Three title attributes can be set.

Attribute	Meaning
`titlecolor`	color of the title
`center`	centers the title
`titlesize`	size of the title

The default alignment is not centered. The keyword `center` allows you to center the title.

Three sizes are available:

- normal-sized title (declaration: `titlesize normal`),
- medium-sized title (declaration: `titlesize medium`),
- large-sized title (declaration: `titlesize large`).

4.2 Text

The model used to declare a text is given below.

```
text
attribute_1 arg_1 arg_2 ...
attribute_2 arg_1 arg_2 ...
...

{line 1 $ }
{line 2 $ }
...
endtext
```

There are two text attributes.

Attribute	Meaning
`textcolor`	color of the text
`textwidth`	width of the text

Every line of the text must ends with a '`$`'. This character is used to force *Visualizer* to break the line. *Visualizer* automatically cuts the line into several lines if the line cannot be printed on a single line.

It is possible to indent a line. To do so, just type:

```
{ _ 5 content of the line $ }
```

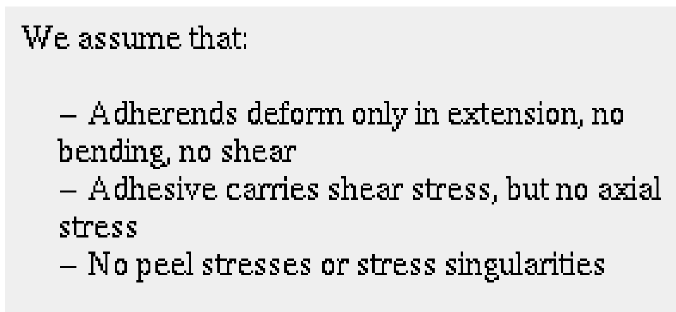
An indented line must begin with a '_'. The number following '_' represents the number of spaces necessary to indent the line. In our example, the number of spaces is 5.

It is also possible to declare items. The model is given below.

```
{ @ 5 content of the line $ }
```

An item must begin with a '@'. The number following '@' represents the number of spaces necessary to indent the item. Contrary to the indented line, if the item cannot be printed on a single line, spaces will be put at the beginning of each line. In our example, the number of spaces is 5.

Here is an example of text:


We assume that:

– Adherends deform only in extension, no bending, no shear
– Adhesive carries shear stress, but no axial stress
– No peel stresses or stress singularities

Figure 7

4.3 Picture

The declaration of a picture is given below.

```
picture
imagename name_of_image
imagewidth width
imageheight height
endpicture
```

The image must be a GIF image. The name of the image must have a GIF extension (.gif).

It is possible to give a number and a caption to the image. If you want to add a caption and a number in our previous example, you just have to type:

```
picture
imagename name_of_image
imagewidth width
imageheight height
figurenumber Figure 1
{ Caption }
endpicture
```

The keyword `figurenumber` sets the number of the image. It will be displayed just at the bottom of the image.

Here is an example of an image with a number (`Figure 1`) and a caption (`Volkerson's analysis`):

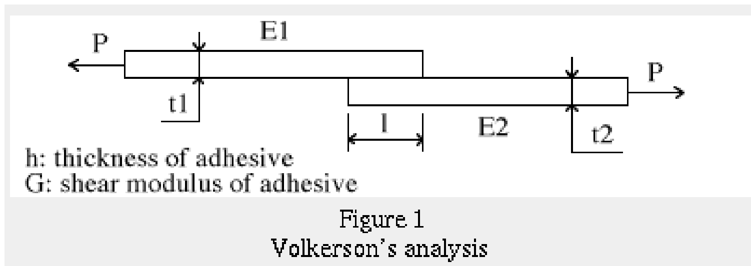


Figure 8

4.4 Navigator

A navigator is a set of buttons which allows the user to navigate in the tutorial.

Figure 9

To declare a navigator, you just have to use the following model:

```
navigator
attribute_1 arg_1 arg_2 ...
attribute_2 arg_1 arg_2 ...
...

{ Button 1 & window_1 }
{ Button 2 & window_2 }
...
endnavigator
```

The two navigator attributes are:

Attribute	Meaning
`buttoncolor`	buttons' color
`labelcolor`	labels' color

The syntax to declare buttons is the same as the one used in the declaration of the main menu.

5 Example

Here is an example of a description file I have written. This file is *description.dat*. It is located in */home3/httpd/htdocs/javaprj/viscprj/visualizer/tutorialchecker*.

```
# File: description.dat
#
# Author: Francois Coupleux (coupleux@eleve.emn.fr)
#
# Date: 7-10-1996


# Default Values

defaultvalues
backgroundcolor 252 253 169
buttoncolor 3 185 237
labelcolor 255 255 255
specialbuttoncolor 253 148 148
speciallabelcolor 255 255 255
maxcontentheight 700
xgap 10
ygap 10
enddefaultvalues

# Main Menu

mainmenu
title
{ Domains }
endtitle
{ Stress Analysis & stressanalysis }
endmainmenu

# Menu for Stress Analysis

menu
name stressanalysis
windowtitle Stress Analysis
{ Joints & joints }
endmenu

# Menu for Joints
```

```
menu
name joints
windowtitle Joints
{ Theory & theory }
{ Assumptions & assumptions }
{ Working Example & example }
endmenu

# Window for the theory

window
name theory
windowtitle Theory for shear lag model
title
titlesize normal
{ Introduction }
endtitle
text
textwidth 600
{ _ 3 The Volkerson's model was ....... $ }
endtext
picture
imagename scheme.gif
imagewidth 400
imageheight 100
figurenumber Figure 1
{ Volkerson's analysis }
endpicture
title
titlesize normal
{ Equation }
endtitle
picture
imagename equation.gif
imagewidth 600
imageheight 200
endpicture
endwindow

# Window for assumptions

window
name assumptions
windowtitle Assumptions for shear lag model
```

```
text
textwidth 300
{ We assume that: $ $ }
{ @ 5 Adherends deform only in extension, no bending, no shear $ }
{ @ 5 Adhesive carries shear stress, but no axial stress $ }
{ @ 5 No peel stresses or stress singularities $ }
endtext
endwindow

# Plotting System for the example

plottingsystem
name example
windowtitle Working example of shear lag model
graphtitle Shear lag model
xtitle x
ytitle tau
xvariable x
yvariable tau
xvariableunit m
equations
{ omega=sqrt((G*(E1*t1+E2*t2))/(h*(E1*E2*t1*t2))) }
{ tau=((P*omega)/(2*sinh(omega*l/2)))*cosh(omega*x)+(P*omega*(E1*t1-E2*t2)*
sinh(omega*x))/(2*(E1*t1+E2*t2)*cosh(omega*l/2)) }
endequations
parameters
{ E1 & Pa }
{ E2 & Pa }
{ t1 & m }
{ t2 & m }
{ l & m }
{ h & m }
{ G & Pa }
{ P & N/m }
endparameters
endplottingsystem
```

Appendices

Appendix A

Visualizer.java

`http://wave.esm.vt.edu/javaprj/viscprj/visualizer/applet/Visualizer.java`

Appendix B

TutorialDescriptionServer.java

`http://wave.esm.vt.edu/javaprj/viscprj/visualizer/applet/TutorialDescriptionServer.java`

Appendix C

PVWaveAccessServer.java

`http://wave.esm.vt.edu/javaprj/viscprj/visualizer/applet/PVWaveAccessServer.java`

Appendix D

tutorialchecker.c

`http://wave.esm.vt.edu/javaprj/viscprj/visualizer/tutorialchecker/tutorialchecker.c`